USC–SIPI REPORT #307

Scalable Photonic Neural Networks for Real-Time Pattern Classification

by

Adam A. Goldstein

May 1997

Signal and Image Processing Institute
UNIVERSITY OF SOUTHERN CALIFORNIA
Department of Electrical Engineering-Systems
3740 McClintock Avenue, Room 404
Los Angeles, CA 90089-2564 U.S.A.

Acknowledgements

I would like to thank, first and foremost, my family for their love and support during my graduate school career. The path to completing this dissertation was far from clear during some difficult times, and I would like to especially thank my mother, father, and sister for their understanding and many kind words of encouragement. I may not have always made it clear enough to them, but their support was crucial to my successful completion of this work.

I would also like to thank my friends and associates at USC, starting with my research advisor, Dr. B. Keith Jenkins. I consider Dr. Jenkins to be not only my advisor, but also a friend with whom I have enjoyed many hours of interesting conversation. Of all the things Dr. Jenkins has taught me over the years about science and engineering, the importance of attention to detail in one's thinking and writing stands out as being the most important. I must admit that Dr. Jenkins' thoroughness was a challenge sometimes, but looking back I now realize that my career will benefit from the experience. Special thanks also go to the other members of my committee, Dr. Sawchuk and Dr. Neumann, as well as to Dr. von der Malsburg, Dr. Willner, and Dr. Pinkston for serving on my qualifying exam committee.

Of course, no acknowledgement section would be complete without due thanks to one's "progress advisor". My friend Clare Waterson was *extremely* diligent in

her role of "kicking me in the behind" in order to be sure that I actually complete this work rather than simply talk about completing it. I would like to thank Clare for all the pleasant "Hi Ad!" greetings over the years and, lest I forget, all of the help with LaTeX.

Thanks also go to the other Ph. D. students in Dr. Jenkins' research group, especially Greg Petrisor with whom I worked most closely and have enjoyed many hours of (sometimes heated!) discussion. I would like to thank Sabino Piazzolla, Ching-Chu Huang, Nainjeet Ramlagan, Yunsong Huang, and Kuang-Yu Li for their technical help and friendship during my graduate career. And last but not least, thanks go to Gloria Halfacre for her consistently cheerful administrative help and her well-stocked supply of gum balls and gummy worms.

Contents

List of Figures

List of Tables

Abstract

With the rapid advancement of photonic technology in recent years, the potential exists for the incorporation of photonic neural-network research into the development of opto-electronic real-time pattern classification systems. In this dissertation we present three classes of photonic neural-network models that were designed to be compatible with this emerging technology: (1) photonic neural networks based upon probability density estimation, (2) photorefractive neural-network models, and (3) vertically stacked photonic neural networks that utilize hybridized CMOS/GaAs chips and diffractive optical elements. In each case, we show how previously developed neural-network learning algorithms and/or architectures must be adapted in order to allow an efficient photonic implementation.

For class (1), we show that conventional "k-Nearest Neighbors" (k-NN) probability density estimation is not suitable for an analog photonic neural-network hardware implementation, and we introduce a new probability density estimation algorithm called "Continuous k-Nearest Neighbors" (C-kNN) that is suitable. For class (2), we show that the diffraction-efficiency decay inherent to photorefractive grating formation adversely affects outer-product neural-network learning algorithms, and we introduce a gain and exposure scheduling technique that resolves the incompatibility. For class (3), the use of compact diffractive optical interconnections constrains the corresponding neural-network weights to be fixed and

locally connected. We introduce a 3-D Photonic Multichip-Module (3-D PMCM) neural-network architecture that utilizes a fixed diffractive optical layer in conjunction with a programmable electronic layer, to obtain a multi-layer neural network capable of real-time pattern recognition tasks. The design and fabrication of key components of the 3-D PMCM neural-network architecture are presented, together with simulation results for the application of detecting and locating the eyes in an image of a human face.

Chapter 1

Introduction

Over the past several years, neural-network technology has begun the transition from being a research curiosity into being a viable technology that can be incorporated into commercial products. Beginning with the increased interest in neural-network research that took place in the mid-1980's, continuing research into topics such as neural-network learning theory, VLSI implementations of neural circuitry, and efficient input data representations have resulted in the development of commercial neural-network products in the mid-1990's. For example, Visionics, Inc. (Metuchen, NJ) now offers a neural-network-based software package for IBM-compatible PC's that authenticates a user through face recognition rather than through a password, using a small camera mounted on top of the PC monitor. Recent neural-network hardware products include digital chips for embedded speech recognition applications (Sensory, Inc., Sunnyvale, CA) and PC co-processor boards for neural-network pattern recognition/image processing operations (Adaptive Solutions, Beaverton, Oregon).

With the rapid advancement of photonic technology in recent years, the potential exists for an analogous incorporation of photonic neural-network research into the development of opto-electronic neural-network products. It has long been recognized that photonic storage media and interconnection systems offer fundamental advantages over their all-electronic counterparts in terms of storage capacity, impedance matching characteristics, and power dissipation requirements [van Heerden, 1963] [Miller, 1989] [Feldman *et al.*, 1988]. However, the device fabrication technology required to realize the potential of photonics for information processing systems such as neural networks has become available only recently. For example, the successful hybridization of CMOS electronics with multiple-quantum-well (MQW) GaAs optical modulators [Goossen *et al.*, 1995] [Kyriakakis *et al.*, 1995] [Worchesky *et al.*, 1996] has enabled the fabrication of dense high-bandwidth planes of artificial neurons that can be used in multi-layer neural-network systems. Progress has also been rapid in the development of optical interconnection components that can implement weighted interconnections between neural planes. It is now possible for an optical engineer to design Diffractive Optical Elements (DOE's) and microlenses [Jahns, 1994] for a neural interconnection system, and have them fabricated using photolithographic processes common in the semiconductor industry.

In this dissertation, we present neural-network algorithms and hardware architectures that were designed to be compatible with this emerging photonic technology. We introduce and discuss three classes of photonic neural-network models: (1) photonic neural networks based upon probability density estimation, (2) photorefractive neural-network models, and (3) vertically stacked photonic neural

networks that utilize hybridized CMOS/GaAs chips and diffractive optical elements. In each case, we show how previously developed neural-network learning algorithms and/or architectures must be adapted in order to allow an efficient photonic implementation. The emphasis of the discussion is towards computationally intensive real-time pattern recognition applications, where a neural network must process large format (*e.g.*, 256 x 256 pixels) input images at a rapid frame rate (*e.g.*, 30 frames/sec). These applications make best use of the high temporal and spatial bandwidth offered by photonic hardware components.

For each of the three neural-network classes discussed above, a photonic hardware implementation imposes important constraints upon the relevant architectures and learning algorithms. In each case, we propose a hardware and/or algorithmic solution that addresses the limitations imposed by a photonic implementation. For class (1), we show that conventional "k-Nearest Neighbors" density estimation is not suitable for an analog photonic neural-network hardware implementation, and we introduce a new density estimation algorithm called "Continuous k-Nearest Neighbors" that is suitable. For class (2), we show that the diffraction-efficiency decay inherent to photorefractive grating formation adversely affects outer-product neural-network learning algorithms, and we introduce a gain and exposure scheduling technique that resolves the incompatibility. For class (3), the use of compact diffractive optical interconnections constrains the corresponding neural-network weights to be fixed and locally connected. We introduce a 3-D Photonic Multichip-Module (3-D PMCM) neural-network architecture that utilizes a fixed diffractive optical layer in conjunction with a programmable electronic layer, to obtain a multi-layer neural network capable of real-time pattern recognition tasks such as locating the eyes in an image of a human face.

The photonic neural networks discussed in this dissertation offer the potential of *scalability* to higher temporal and spatial bandwidths than are possible with purely electronic neural networks. Temporal bandwidth refers to the rate at which the neural input and output signals can be propagated through the network. Temporal-bandwidth scalability is achieved in the photonic neural-network models discussed above through the use of GaAs modulators and free-space optical interconnections, which allow high-speed (*e.g.*, 250 MHz) neural signal modulation that is independent of the distance between neural-network layers [Miller, 1989]. Spatial bandwidth refers to the number of neurons that can be packed into each layer. For classes (1) and (2) discussed above, spatial-bandwidth scalability is achieved through the use of volume holographic storage of interconnection weights, which allows a greater connectivity between planes of closely-packed neurons than would be possible with planar-electronic interconnections [Psaltis *et al.*, 1988]. For class (3), spatial-bandwidth scalability is achieved using the 3-D PMCM architecture, which bypasses the usual 2-D chip-to-chip Input/Output (I/O) bottleneck [Goodman, 1984] by allowing optical propagation perpendicular to the neural planes. Thus, our overall goal is to formulate and solve systems-level design problems that are relevant to several varieties of scalable photonic neural networks.

1.1 Contributions

The original contributions contained in each chapter of this dissertation can be summarized by the following:

1. *Continuous k-Nearest Neighbor (C-kNN) algorithm.* We present a new probability density estimation algorithm that was specifically designed for analog electronic or photonic neural-network implementation. A formal proof of statistical consistency is given for the new probability density estimation algorithm, along with empirical simulation results that suggest performance improvements with respect to conventional k-NN and Parzen Windows techniques. Bayesian pattern classification and nonparametric regression architectures that use C-kNN probability density estimation are presented.

2. *Photonic neural-network architectures for C-kNN and other probabilistic neural networks.* Volume and planar holographic architectures for neural networks that are based upon nonparametric probability density estimation algorithms are presented. Photonic architectures for neural-network implementations of probability density estimation, Bayesian pattern classification, and nonparametric regression are given. We present derivations for the optical encoding of bipolar training vectors and the relationship between spatial-light-modulator (SLM) gain and estimation kernel width.

3. *Gain and exposure scheduling.* We present a gain and exposure scheduling technique that compensates for photorefractive diffraction-efficiency decay when applied to outer-product neural-network learning algorithms. A derivation that maps the photorefractive grating update equations into neural-network weight updates is presented, along with empirical simulation results which verify that our method works correctly when applied to the Exclusive-OR (XOR) problem. It is shown that the method applies to both Single

Coherent Source (SCS) and the Incoherent/Coherent (I/C) optical representation.

4. *A 3-D Photonic Multichip-Module (3-D PMCM) neural-network architecture.* The hardware architecture for a 3-D stack of opto-electronic devices used for feedforward neural-network processing is presented. The VLSI design, computer simulation results, and electrical test results for the opto-electronic chip that provides a scrolling-window input to the neural-network processor are given. Derivations for the mapping between neural-network signals and 3-D PMCM optical representations are given. We present examples of fabricated micro-diffractive optical elements designed for neural-network edge detection, and we derive optical imaging constraints which show that the 3-D PMCM optical components can be stacked compactly.

5. *Eye-detection simulations for a 3-D PMCM neural network.* A two-layer neural network that is consistent with the 3-D PMCM hardware architecture is simulated for the application of detecting the positions of the two eyes in an image of a human face. Space-variant edge detection, which requires only local neural-network connectivity, is shown to be a useful pre-processing stage for eye detection. Simulation results on a database of real facial images show significant performance improvements compared to a simpler single-layer network. We show that the two-layer neural-network learning algorithm is computationally intensive when implemented on a digital computer (20 seconds per image on a SPARC 10 workstation), while the 3-D PMCM neural network can potentially process the images in real time (1/30 second per image).

1.2 Organization

The five technical chapters of this dissertation (chapters 2-6) are organized into 3 parts, where each part corresponds to one of the 3 classes of neural-network models discussed above. Chapters 2-3 discuss neural networks based upon probability density estimation. In Chapter 2 the C-kNN density estimation algorithm is introduced from a theoretical viewpoint, while Chapter 3 builds on the theoretical foundation established in Chapter 2 with a discussion of several photonic implementations of probabilistic neural networks. The second part of the dissertation consists of Chapter 4, which discusses the gain and exposure scheduling technique for photorefractive neural network learning algorithms. The final part of this dissertation consists of chapters 5-6, which discuss the 3-D PMCM Neural Network architecture. The 3-D PMCM hardware is discussed first, in Chapter 5, followed by the eye-detection simulation results presented in Chapter 6. Finally, Chapter 7 concludes with a summary and a discussion of future work.

Chapter 2

Probability density estimation for analog neural network implementation: the Continuous k-Nearest Neighbors Algorithm

2.1 Introduction

The k-Nearest Neighbors (k-NN) algorithm [Fix and Hodges, 1951] is one of the most popular and most researched classification techniques in the field of statistical pattern recognition [Dasarathy, 1991] [Fukunaga, 1990]. It is based on the intuitively simple idea that a good estimate for the classification of an unknown input vector is the class most represented by its nearest neighbors. However, this algorithm has always had two limitations preventing its widespread use in real-time pattern recognition systems: (1) the algorithm requires a memory capacity large enough to store all of the necessary training vectors, and (2) the algorithm requires the computation of the distance between the unknown input vector and each of the training vectors, which implies an increasing computational time-complexity as the number of training vectors grows. In the past, the issues

cited above have been addressed in the statistical pattern recognition literature [Dasarathy, 1991]. Algorithmic advances have been achieved that in some cases allow a reduced number of training vectors to be stored, thus partially alleviating problem (1). Likewise, problem (2) has been addressed by the development of fast searching techniques to find the nearest neighbors.

More recently, advances in analog electronic and opto-electronic implementations of artificial neural networks have provided possible hardware solutions to the problems discussed above. For example, issue (2) has been addressed by Urahama *et al.*, who have designed special-purpose analog neural network circuits to implement the k-NN algorithm using sub-threshold MOS transistors [Urahama and Nagao, 1994]. As will be discussed in more detail below, recent research efforts in the analog electronic implementation of radial basis function (RBF) neural networks [Sheu and Choi, 1995] can also be applied to the k-NN algorithm. Progress in opto-electronic hardware implementations of neural network architectures has the potential of directly addressing problem (1). In particular, the use of volume holographic materials in neural network architectures [Psaltis *et al.*, 1988] [Asthana *et al.*, 1993] [Li *et al.*, 1993] can theoretically provide a storage capacity as high as 10^{12} analog values / cm^3.

The theory underlying the k-NN classification algorithm is based on nonparametric probability density estimation. Once the underlying probability densities that describe a data set are known, it is possible to design a statistically optimal (Bayes) classifier. For the case of k-NN classification, the underlying densities are approximated from the training data using k-NN density estimation [Loftsgaarden and Quesenberry, 1965], and the density estimates are then used to construct a Bayes classifier. In this chapter, we describe a new probability

density estimation algorithm called "Continuous-k Nearest Neighbors" (C-kNN). This algorithm is specifically designed to be implemented using electronic or opto-electronic analog neural networks of the type described above.

This chapter is organized as follows. Section 2.2 begins with a brief review of conventional k-NN and Parzen windows density estimation, and it establishes the notation used for the remainder of the chapter. Section 2.3 introduces C-kNN probability density estimation. We present analytic and empirical results to verify that the C-kNN probability density estimate converges in probability to the true underlying density. In Section 2.4, the performance of the C-kNN technique is compared to that of conventional Parzen windows and k-NN. We show that, for some classes of density estimation problems, the C-kNN algorithm performs better than both Parzen windows and k-NN. The remainder of this chapter focuses on neural network implementations and applications for C-kNN density estimation. In Section 2.5, radial basis function neural networks for probability density estimation are presented, and in Section 2.6 these RBF neural networks are discussed in the context of C-kNN pattern classification and nonparametric regression. The chapter concludes with a summary and discussion in Section 2.7.

2.2 Review of Parzen windows and conventional k-NN density estimation

The probability density estimation problem can be defined as follows: given a set of n independent identically distributed ($i.i.d.$) training vectors $\mathbf{T} = \{\mathbf{t}^{(i)} : i = 1, 2, \cdots, n\}$ drawn from a population with probability density function $p(\mathbf{x})$, find an estimate $\hat{p}_n(\mathbf{x}; \mathbf{T})$ that approximates the true density $p(\mathbf{x})$. (The functional

dependence of $\hat{p}_n(\mathbf{x})$ on $\mathbf{T}$ is suppressed when there is no ambiguity) The density estimate is considered nonparametric if there is no assumed functional form for the true density. This paper will discuss only nonparametric density estimation.

The kernel interpolation technique known as Parzen windows is commonly used for nonparametric density estimation [Rosenblatt, 1962] [Parzen, 1962] [Cacoullos, 1964]. The Parzen window estimate is defined by the expression

$$\hat{p}_n(\mathbf{x}) = \frac{k_n(\mathbf{x})/n}{V_n}, \tag{2.1}$$

in which

$$k_n(\mathbf{x}) = \sum_{i=1}^{n} \phi\left(\frac{\mathbf{x} - \mathbf{t}^{(i)}}{h_n}\right) \tag{2.2}$$

and $\phi(\cdot)$ is the estimation kernel. The kernel volume V_n is related to the kernel width h_n by the relation

$$\begin{aligned} V_n &= \int_{\mathbf{x}'} \phi(\mathbf{x}'/h_n)\, d\mathbf{x}' \\ &= h_n^d \int_{\mathbf{x}'} \phi(\mathbf{x}') d\mathbf{x}' \\ &= h_n^d V_0, \end{aligned} \tag{2.3}$$

where d is the number of components in the input vector $\mathbf{x}$, and the unscaled kernel volume V_0 is defined above. The kernel $\phi(\cdot)$ must satisfy the following conditions: the integral in Eq. (2.3) must exist, the kernel must be nonnegative and bounded, and it must satisfy the limit

$$\lim_{|\mathbf{x}|\to\infty} \phi(\mathbf{x}) \Pi_{i=1}^{d} x_i = 0, \tag{2.4}$$

in which || denotes the Euclidean vector magnitude. It can be shown that if the deterministic sequence $\{h_n\}$ is chosen such that $h_n \to 0$ and $nh_n^d \to \infty$ as $n \to \infty$ (*e.g.*, $h_n \propto 1/\sqrt{n}$), then the estimated density converges in probability to the true density [Duda and Hart, 1973].

Another commonly used nonparametric technique is k-NN density estimation [Fix and Hodges, 1951] [Loftsgaarden and Quesenberry, 1965]. This estimate can be defined as

$$\hat{p}_n^{(\mathrm{K})}(\mathbf{x}) = \frac{k_n/n}{V_n^{(\mathrm{K})}(\mathbf{x})}, \tag{2.5}$$

in which the (K) superscript denotes k-NN density estimation. In contrast with Parzen window estimation, here the kernel volume $V_n^{(\mathrm{K})}(\mathbf{x})$ is a random variable dependent on the input vector $\mathbf{x}$ and the training set $\mathbf{T}$, while $\{k_n\}$ is a deterministic integer sequence chosen *a priori*. The volume $V_n^{(\mathrm{K})}(\mathbf{x})$ is determined by the Euclidean distance between the input vector $\mathbf{x}$ and its k_n^{th} nearest neighbor among the set of training vectors $\mathbf{T}$. If this distance is denoted $r_n(\mathbf{x})$, then $V_n(\mathbf{x})$ is equal to the multi-dimensional spherical volume [Loftsgaarden and Quesenberry, 1965]

$$V_n^{(\mathrm{K})}(\mathbf{x}) = \frac{2r_n^d(\mathbf{x})\pi^{d/2}}{d\Gamma(d/2)}, \tag{2.6}$$

where $\Gamma(\cdot)$ is the Gamma function defined by $\Gamma(g) \triangleq \int_0^\infty x^{g-1}e^{-x}dx$. It can be shown that if the sequence $\{k_n\}$ is chosen such that $k_n \to \infty$ and $k_n/n \to 0$ as $n \to \infty$, (*e.g.*, $k_n \propto \sqrt{n}$), then the estimated density converges in probability to the true density [Loftsgaarden and Quesenberry, 1965].

2.3 C-kNN probability density estimation

It is well known that the k-NN method has several drawbacks. For example, it uses a hard threshold to decide whether or not a given training vector contributes to the density estimate: if the distance between $\mathbf{x}$ and $\mathbf{t}^{(i)}$ is larger than the distance between $\mathbf{x}$ and its $k^{\underline{\text{th}}}$ nearest neighbor, then $\mathbf{t}^{(i)}$ is completely ignored in the calculation of the density estimate [Buturovic, 1993]. In addition, the k-NN method treats all training vectors that are sufficiently close to the input vector identically; that is, the nearest neighbor to $\mathbf{x}$ has the same contribution to the density estimate as the $k^{\underline{\text{th}}}$ nearest neighbor [Mack and Rosenblatt, 1979] [Dudani, 1976] [Silverman, 1986]. On the other hand, although the Parzen windows method does not have the aforementioned drawbacks, the method is limited because the kernel width is fixed for all values of the input vector. This limitation can be a problem if the density being estimated is more efficiently approximated by a kernel that can "adapt" to the local probability density [Silverman, 1986].

Many of these issues have been previously addressed in the literature. For instance, Dudani proposed a "weighted-distance k-NN" rule that explicitly gives nearby training samples more weight in the classification decision than those that are more distant [Dudani, 1976]. The effectiveness of this method has been actively debated since its introduction [Baily and Jain, 1978] [Keller *et al.*, 1985] [Macleod *et al.*, 1987] [Denoeux, 1995]. Modifications to the Parzen windows method have also been reported in the literature, in order to address the problem of fixed-width kernels as mentioned above [Silverman, 1986]. In this paper, our approach is to combine the Parzen windows and k-NN density estimation techniques. The new estimation technique, called "Continuous k-Nearest Neighbors"

(C-kNN), retains both the continuous kernel property of Parzen windows and the adaptive kernel property of k-NN.

2.3.1 Definition of C-kNN

As was the case with k-NN, the C-kNN method uses a kernel volume that is a function of the input vector $\mathbf{x}$:

$$\hat{p}_n^{(\mathrm{C})}(\mathbf{x}) = \frac{k_n^{(\mathrm{C})}/n}{V_n^{(\mathrm{C})}(\mathbf{x})}, \tag{2.7}$$

where the superscript (C) denotes C-kNN density estimation. Here, the kernel volume $V_n^{(\mathrm{C})}(\mathbf{x})$ is defined more generally than was the case for k-NN:

$$V_n^{(\mathrm{C})}(\mathbf{x}) = \int_{\mathbf{x}'} \phi\left(\mathbf{x}'/h_n(\mathbf{x})\right) d\mathbf{x}' = h_n^d(\mathbf{x}) V_0, \tag{2.8}$$

in which the kernel width $h_n(\mathbf{x})$ is chosen to satisfy

$$\sum_{i=1}^{n} \phi\left(\frac{\mathbf{x} - \mathbf{t}^{(i)}}{h_n(\mathbf{x})}\right) = k_n^{(\mathrm{C})}. \tag{2.9}$$

Eq. (2.9) defines a feedback mechanism in which the kernel width $h_n(\mathbf{x})$ is varied until the summation of all kernels is equal to a predetermined parameter, $k_n^{(\mathrm{C})}$. If Eq. (2.9) does not uniquely determine the kernel width $h_n(\mathbf{x})$ [as may be the case if $\phi(\cdot)$ is constant in some regions], then the width is defined to be the minimum value consistent with Eq. (2.9). The parameter $k_n^{(\mathrm{C})}$ is similar to k_n in k-NN density estimation, with the exception that $k_n^{(\mathrm{C})}$ can in general take on any real value, rather than being restricted to integer values as is the case with k-NN. The estimation kernel $\phi(\cdot)$ used for C-kNN density estimation must satisfy the same

conditions previously discussed for Parzen windows. In addition, the kernel must satisfy

$$\phi(\mathbf{x}) > 0 \quad \forall |\mathbf{x}| \leq r \tag{2.10}$$

for some $r > 0$. That is, the kernel must include a finite region surrounding the origin, $\mathbf{x} = 0$.

By comparing Eq. (2.7) with Eq. (2.5), and Eq. (2.8) with Eq. (2.3), it is clear that C-kNN is a combination of Parzen windows and k-NN. C-kNN uses a continuous kernel $\phi(\cdot)$, as does Parzen windows, but the volume $V_n^{(\mathrm{C})}(\mathbf{x})$ is a function of the input vector, as is the case with k-NN. For the special case

$$\phi(\mathbf{x}) = \begin{cases} 1 & \text{if } |\mathbf{x}| \leq 1 \\ 0 & \text{otherwise} \end{cases}, \tag{2.11}$$

the parameter $k_n^{(\mathrm{C})}$ must be an integer in order for Eqs. (2.9) and (2.11) to be mutually consistent. Furthermore, Eqs. (2.9) and (2.11) require the kernel width $h_n(\mathbf{x})$ to be chosen as the distance to the $k_n^{(\mathrm{C})}$ $\underline{\text{th}}$ nearest neighbor. Therefore, in this special case the RHS of Eq. (2.7) reduces to the RHS of Eq. (2.5) (where $k_n^{(\mathrm{C})}$ takes the place of k_n), and the definition of $V_n^{(\mathrm{C})}(\mathbf{x})$ becomes equivalent to the definition of $V_n^{(\mathrm{K})}(\mathbf{x})$ [where $h_n(\mathbf{x})$ takes the place of $r_n(\mathbf{x})$, and $V_0 = 2\pi^{d/2}/\left(d\Gamma(d/2)\right)$]. That is, in this special case C-kNN becomes equivalent to k-NN.

2.3.2 Proof of C-kNN consistency

A probability density estimation algorithm is consistent if $\hat{p}_n(\mathbf{x}) \rightarrow_{\mathrm{P}} p(\mathbf{x})$, where the symbol $\rightarrow_{\mathrm{P}}$ denotes stochastic convergence in probability. That is, the density

estimate is consistent if [Papoulis, 1965]

$$\forall \mathbf{x}, \lim_{n\to\infty} P(|\hat{p}_n(\mathbf{x}) - p(\mathbf{x})| > \epsilon) = 0 \quad \forall \epsilon > 0. \tag{2.12}$$

As mentioned above, both Parzen windows and k-NN are consistent density estimates if the $\{h_n\}$ (Parzen windows) or $\{k_n\}$ (k-NN) sequences are properly chosen. In the following, we show that C-kNN is also a consistent density estimate.

Assume $k_n^{(C)}$ is chosen such that the following two conditions are met:

$$\lim_{n\to\infty} k_n^{(C)} = \infty \tag{2.13}$$

$$\lim_{n\to\infty} (k_n^{(C)}/n) = 0. \tag{2.14}$$

In the following, we prove that C-kNN is consistent by relating the above two conditions to the asymptotic behavior of the kernel width $h_n(\mathbf{x})$. For any fixed input vector $\mathbf{x}$, the infinite random sequence $\{h_n(\mathbf{x})\}$ is related to the deterministic sequence $\{k_n^{(C)}\}$ through Eq. (2.9). To prove that the C-kNN estimate converges correctly, we will show that the sequence $\{h_n(\mathbf{x})\}$ satisfies the two Parzen window convergence conditions: $h_n(\mathbf{x}) \to_P 0$ (Sec. 2.3.2), and $nh_n^d(\mathbf{x}) \to_P \infty$ (Sec. 2.3.2).

First Parzen convergence condition

Using Eq. (2.9),

$$(1/n)\sum_{i=1}^{n} \phi\left(\frac{\mathbf{x} - \mathbf{t}^{(i)}}{h_n(\mathbf{x})}\right) = (k_n^{(C)}/n). \tag{2.15}$$

If the samples $\{\mathbf{t}^{(i)}\}$ are *i.i.d.* random vectors, then

$$(1/n)\sum_{i=1}^{n}\phi\left(\frac{\mathbf{x}-\mathbf{t}^{(i)}}{h_n(\mathbf{x})}\right) \rightarrow_{\mathrm{P}} E_{\mathbf{t}'}\left\{\phi\left(\frac{\mathbf{x}-\mathbf{t}'}{h_n(\mathbf{x})}\right)\right\} = \int_{\mathbf{t}'} p(\mathbf{t}')\phi\left(\frac{\mathbf{x}-\mathbf{t}'}{h_n(\mathbf{x})}\right) d\mathbf{t}'. \quad (2.16)$$

Using Eq. (2.14) and the RHS of Eq. (2.16), the kernel width $h_n(\mathbf{x})$ must satisfy the following conditions $\forall \mathbf{x}$:

$$\int_{\mathbf{t}'} p(\mathbf{t}')\phi\left(\frac{\mathbf{x}-\mathbf{t}'}{h_n(\mathbf{x})}\right) d\mathbf{t}' \rightarrow_{\mathrm{P}} 0, \text{ which implies that} \quad (2.17)$$

$$\int_{\mathcal{R}(\mathbf{x})} p(\mathbf{t}')\phi\left(\frac{\mathbf{x}-\mathbf{t}'}{h_n(\mathbf{x})}\right) d\mathbf{t}' \rightarrow_{\mathrm{P}} 0, \text{ where } \mathcal{R}(\mathbf{x}) = \left\{\mathbf{t}' : \left|\frac{\mathbf{x}-\mathbf{t}'}{h_n(\mathbf{x})}\right| < r'\right\} \quad (2.18)$$

Eq. (2.18) follows from Eq. (2.17) because it represents an integration over a subset of the entire region, in which the integrand is nonnegative. If $p(\mathbf{t}') > 0$ over an infinitesimal region containing $\mathbf{x}$, then $\exists r' \leq r$ such that $\forall \mathbf{t}' \in \mathcal{R}(\mathbf{x})$: $p(\mathbf{t}') > p_m > 0$ and $\phi(\frac{\mathbf{x}-\mathbf{t}'}{h_n(\mathbf{x})}) > \phi_m > 0$ (from Eq. (2.10)). Therefore, the volume of the spherical region $\mathcal{R}(\mathbf{x}) \rightarrow_{\mathrm{P}} 0$ because the integrand is bounded from below by a fixed positive constant, $p_m\phi_m$. Thus, $h_n(\mathbf{x}) \rightarrow_{\mathrm{P}} 0$ because r' is fixed.

If $p(\mathbf{t}') = 0$ over a finite region containing $\mathbf{x}$, then $h_n(\mathbf{x}) \rightarrow_{\mathrm{P}}$ (non-zero constant), which violates the first Parzen convergence condition. However, in this case $\hat{p}_n^{(\mathrm{C})}(\mathbf{x}) \rightarrow_{\mathrm{P}} 0 = p(\mathbf{x})$ (from Eq. (2.7), because $(k_n^{(\mathrm{C})}/n) \rightarrow 0$ and $V_n^{(\mathrm{C})}(\mathbf{x})$ is finite), which means that the estimate converges to the correct value even though $h_n(\mathbf{x}) \not\rightarrow_{\mathrm{P}} 0$.

Second Parzen convergence condition

Using the above result that $h_n(\mathbf{x}) \rightarrow_{\mathrm{P}} 0\ \forall \mathbf{x}$ such that $p(\mathbf{x}) \neq 0$, we now show that $E\{\hat{p}_n^{(\mathrm{C})}(\mathbf{x})\} \rightarrow_{\mathrm{P}} p(\mathbf{x})\ \forall \mathbf{x}$ such that $p(\mathbf{x}) \neq 0$. This preliminary result will then be

used to show that $nh_n^d(\mathbf{x}) \rightarrow_{\mathrm{P}} \infty$, which will complete the proof that the C-kNN estimate is consistent. From the definition of $\hat{p}_n^{(\mathrm{C})}(\mathbf{x})$ given in Eqs. (2.7) and (2.9),

$$\begin{aligned} E\{\hat{p}_n^{(\mathrm{C})}(\mathbf{x})\} &= (1/n)\sum_{i=1}^{n} E_{\mathbf{t}^{(i)}}\left\{\phi\left(\frac{\mathbf{x}-\mathbf{t}^{(i)}}{h(\mathbf{x})}\right)/V_n^{(\mathrm{C})}(\mathbf{x})\right\} \\ &= E_{\mathbf{t}'}\left\{\phi\left(\frac{\mathbf{x}-\mathbf{t}'}{h(\mathbf{x})}\right)/V_n^{(\mathrm{C})}(\mathbf{x})\right\} \end{aligned} \tag{2.19}$$

assuming *i.i.d.* training set vectors $\{\mathbf{t}^{(i)}\}$. As $h_n(\mathbf{x}) \rightarrow_{\mathrm{P}} 0$,

$$\begin{aligned} E\{\hat{p}_n^{(\mathrm{C})}(\mathbf{x})\} &\rightarrow_{\mathrm{P}} E_{\mathbf{t}'}\{\delta(\mathbf{x}-\mathbf{t}')\} \\ &= \int_{\mathbf{t}'} p(\mathbf{t}')\delta(\mathbf{x}-\mathbf{t}')d\mathbf{t}' \\ &= p(\mathbf{x}), \end{aligned} \tag{2.20}$$

where $\delta(\cdot)$ represents the Dirac delta function. Taking the expected value of Eq. (2.7) (noting that $k_n^{(\mathrm{C})}$ is deterministic) and comparing the result with Eq. (2.20), leads to the limit

$$k_n^{(\mathrm{C})} E\left\{1/(nV_n^{(\mathrm{C})}(\mathbf{x}))\right\} \rightarrow_{\mathrm{P}} p(\mathbf{x}), \tag{2.21}$$

which implies that

$$E\left\{1/(nV_n^{(\mathrm{C})}(\mathbf{x}))\right\} \rightarrow_{\mathrm{P}} 0 \tag{2.22}$$

because $k_n^{(\mathrm{C})} \rightarrow \infty$ (Eq. (2.13)) and $p(\mathbf{x})$ is assumed to be finite. Because $(1/(nV_n^{(\mathrm{C})}(\mathbf{x})))$ is a non-negative random variable, Eq. (2.22) implies that $(1/(nV_n^{(\mathrm{C})}(\mathbf{x}))) \rightarrow_{\mathrm{P}} 0$. Therefore $nV_n^{(\mathrm{C})}(\mathbf{x}) \rightarrow_{\mathrm{P}} \infty$, which implies that $nh_n^d(\mathbf{x}) \rightarrow_{\mathrm{P}} \infty$. □

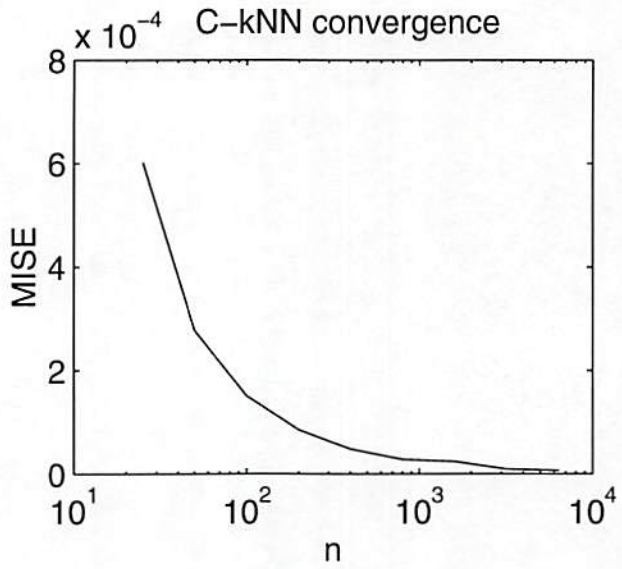


Figure 2.1: C-kNN density estimation convergence as a function of sample size n

2.3.3 Empirical validation of C-kNN consistency

The C-kNN method was tested empirically using a univariate Gaussian distribution N(m=7, σ=4) as the true density, where m represents the Gaussian mean and σ represents the Gaussian standard deviation. The estimation kernel was also chosen to be Gaussian shaped (standard choice for nonparametric density estimation), as given by $\phi(x) = \exp(-x^2/2)$. We chose the $\{k_n^{(C)}\}$ sequence to be $k_n^{(C)} = (0.5)n^{4/5}$, which satisfies the two conditions for consistency [Eqs. (2.13) and (2.14)]. The exponent value (4/5) has been shown to be optimal for estimating Gaussian densities using the conventional k-NN method [Silverman, 1986] (although this does not necessarily imply that (4/5) is optimal for C-kNN).

Fig. 2.1 shows the results of statistically averaged experiments using the parameters given above. The figure of merit used to measure the accuracy of the density estimate is the mean integrated squared error [Silverman, 1986], as given

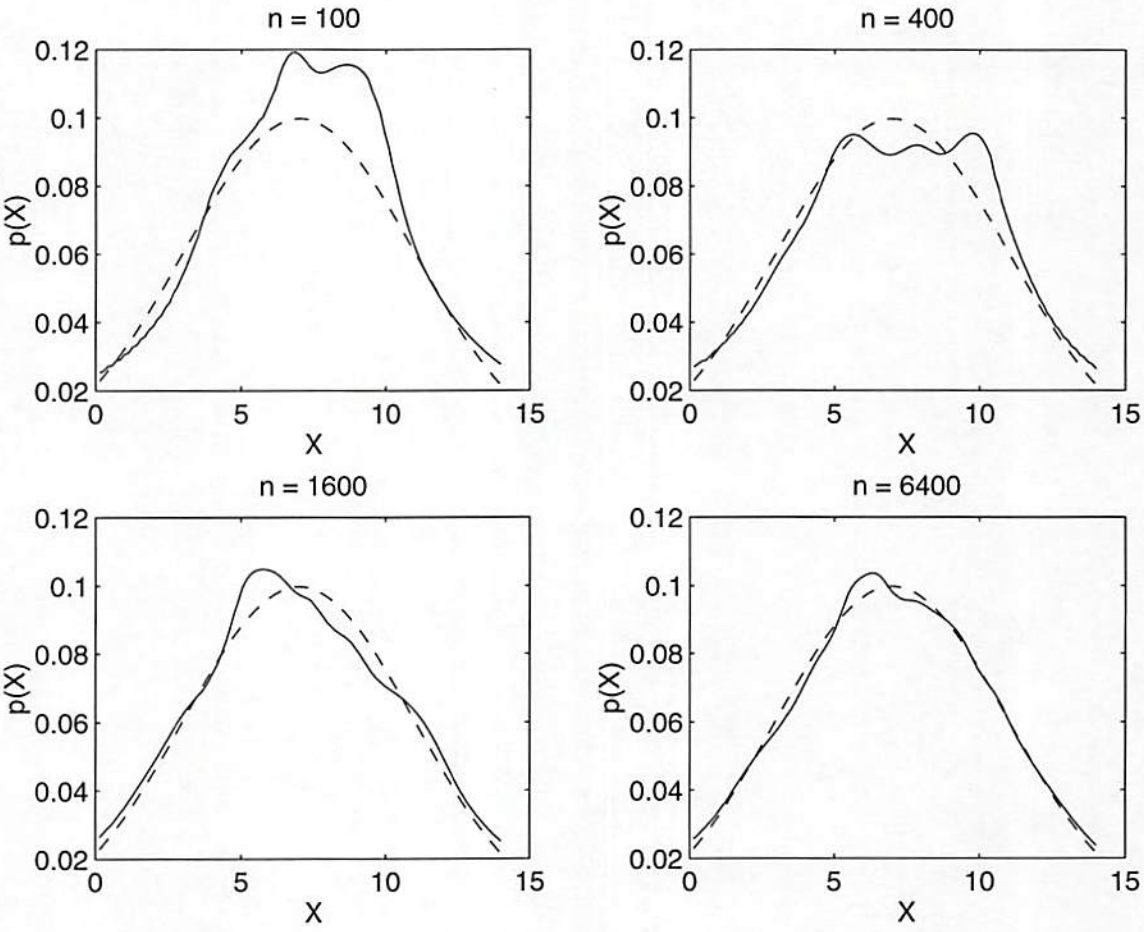


Figure 2.2: C-kNN density estimation for variable sample size n. Dashed line represents true density, solid line represents estimated density. Mean Integrated Squared Error results: $n = 100$, MISE=0.14×10^{-3}; $n = 400$, MISE=0.048×10^{-3}; $n = 1600$, MISE=0.025×10^{-3}; $n = 6400$, MISE=$.0062 \times 10^{-3}$.

by

$$\text{MISE}(\hat{p}_n^{(C)}) = (1/T)\sum_{i=1}^{T}\int_{\mathbf{x}'}[\hat{p}_{n,i}^{(C)}(\mathbf{x}') - p(\mathbf{x}')]^2 d\mathbf{x}', \tag{2.23}$$

where $\hat{p}_{n,i}^{(C)}(\mathbf{x}')$ denotes the estimated density for trial number i given n training samples, and T is the number of independent trials used for ensemble averaging. As indicated in Eq. (2.23), the quantity MISE is itself a random variable, and it is dependent on sample size T. In order to assess the statistical sampling error in the measurement of MISE, we estimate the standard deviation σ_{MISE} indirectly, using [Papoulis, 1965]

$$\sigma_{\text{MISE}} \approx (1/\sqrt{T})\ \text{Sample Std. Dev.}\{\int_{\mathbf{x}'}[\hat{p}_{n,i}^{(C)}(\mathbf{x}') - p(\mathbf{x}')]^2 d\mathbf{x}'\}, \tag{2.24}$$

where *i.i.d.* trials have been assumed. For $T = 50$ trials, the percentage error $\sigma_{\text{MISE}}/\text{MISE} < 15\%$ at all data points. As shown in the figure, the MISE approaches zero as n becomes large. This convergence is shown graphically in Fig. 2.2, where typical examples of true and estimated density functions are shown for increasing sample size n.

2.4 Density estimation performance comparison

The performance of the C-kNN method was assessed by comparing the MISE of the C-kNN estimate to those of the Parzen windows estimate and the k-NN estimate. A mixture density consisting of the sum of two gaussians (7/8)N(m=7, σ=4) + (1/8)N(m=7, σ=.5) was used as the true density to be estimated, and $n = 100$ training samples were drawn from this mixture density for each trial. These mean

and standard deviation parameters were chosen so that the local spatial-frequency content of the true density varies significantly over the x-axis. Gaussian kernels $[\phi(x) = \exp(-x^2/2)]$ were used for both the Parzen windows and C-kNN methods.

The performance of each of the three methods depends on one smoothing parameter: $k^{(C)}$ for C-kNN, k for k-NN, and h for Parzen windows. Therefore, each method was tested using a range of values for the relevant parameter, and the minimum value of the MISE served as the performance metric. Figures 2.3-2.5 show the results of the performance comparison. The upper plot in each figure shows the MISE as a function of the relevant smoothing parameter. Statistically averaging over $T = 100$ trials resulted in a percentage error $\sigma_{\text{MISE}}/\text{MISE} < 10\%$ at all data points along each curve. As shown in the figures, C-kNN performed the best and Parzen windows performed the worst for the mixture density we tested.

Using the optimal parameter values obtained from the upper plots, typical examples of the true and estimated densities are shown in the lower plots. For these examples, sets of 100 samples were repeatedly drawn until a single set gave typical results for all three methods simultaneously (where "typical" is defined to be an integrated squared error within 10% of the MISE). From the lower plot in Fig. 2.3, it is evident that Parzen windows has poor performance because it is unable to adequately model both the high frequency (center) and low frequency (tails) portions of the mixture density. The k-NN method does not suffer from the above weakness (because the kernel width adapts to the local density), but it is clear from Fig. 2.4 that the method yields an estimate that is very "jagged" (*i.e.*, it has a discontinuous first derivative). This effect can be traced back to the hard-threshold nature of the k-NN density estimation algorithm [Buturovic, 1993]. The C-kNN method has better performance than either of the conventional methods

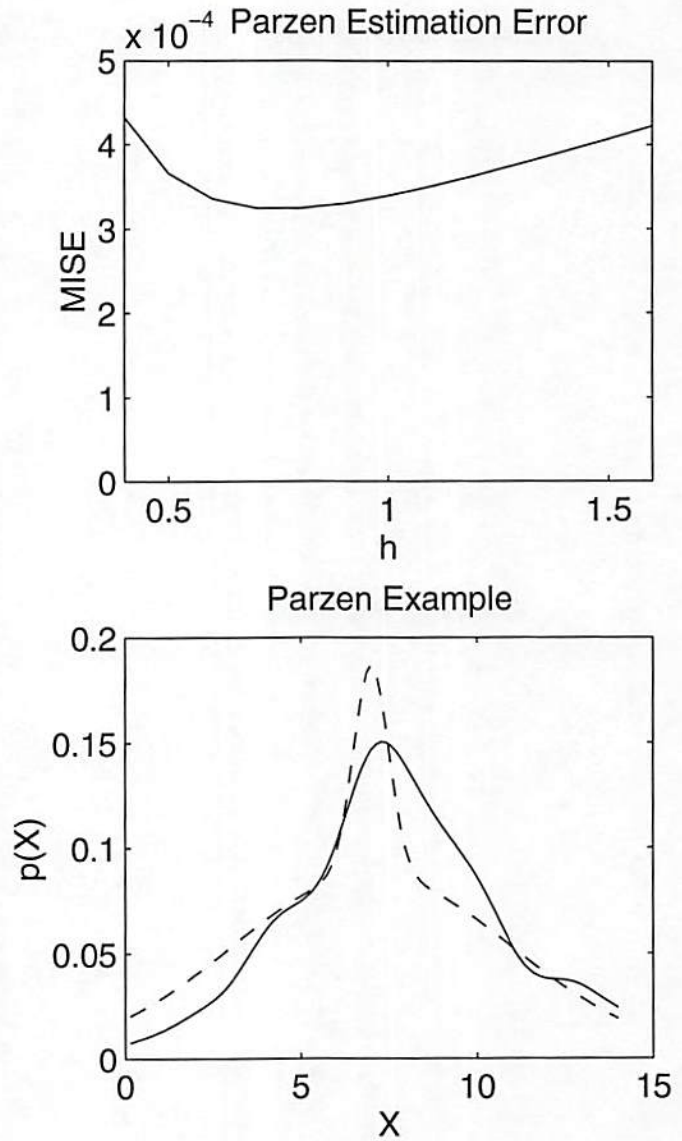


Figure 2.3: Parzen windows performance. The upper plot shows the optimization of h with respect to the performance metric MISE ($h = 0.8$, MISE=3.25×10^{-4}). The lower plot shows a typical example (MISE=3.48×10^{-4}) of true density (dashed) compared to estimated density (solid) using $h = 0.8$.

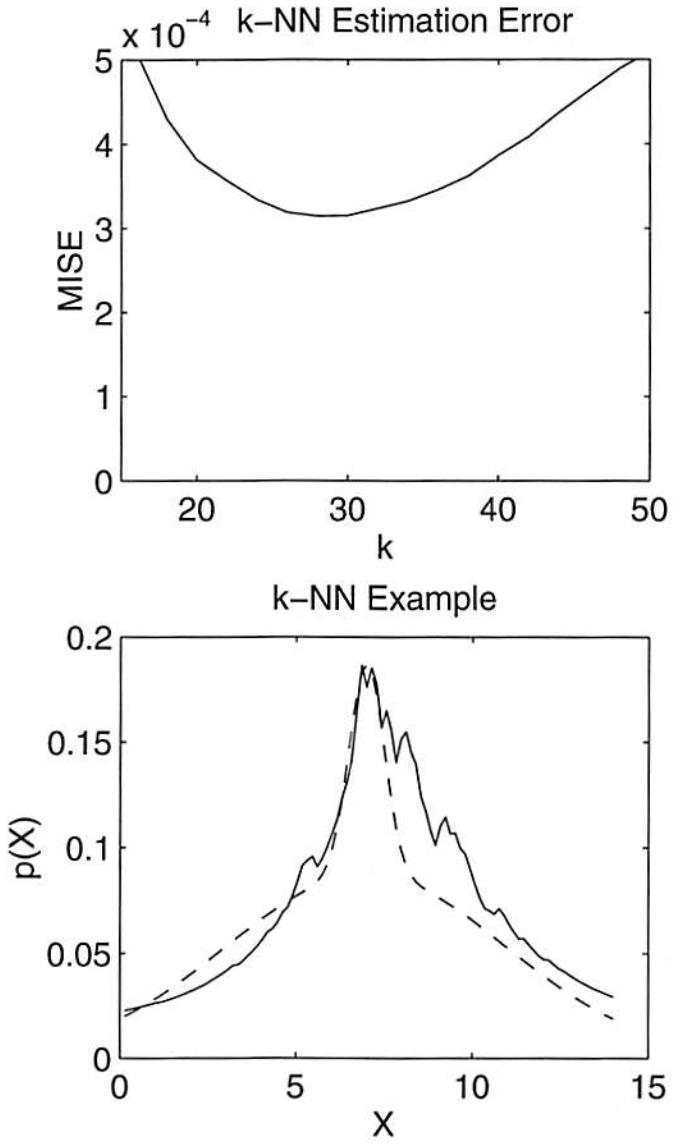


Figure 2.4: k-NN performance. The upper plot shows the optimization of k with respect to the performance metric MISE ($k = 28$, MISE=3.14×10^{-4}). The lower plot shows a typical example (MISE=3.45×10^{-4}) of true density (dashed) compared to estimated density (solid) using $k = 28$.

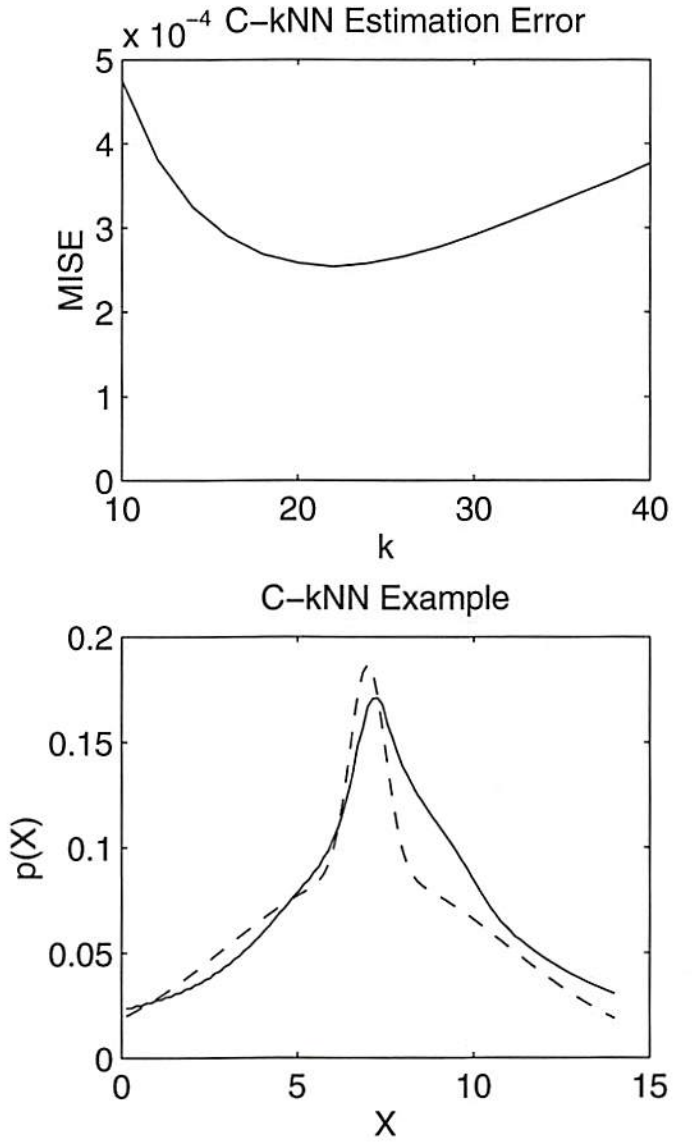


Figure 2.5: C-kNN performance. The upper plot shows the optimization of $k^{(C)}$ with respect to the performance metric MISE ($k^{(C)} = 22$, MISE=2.54×10^{-4}). The lower plot shows a typical example (MISE=2.69×10^{-4}) of true density (dashed) compared to estimated density (solid) using $k^{(C)} = 22$.

because it uses a continuous kernel, which leads to a smooth density estimate, but the kernel width than can still adapt to the local probability density.

2.5 RBF neural-network implementation of nonparametric density estimation

The computation of Eqs. (2.2) and (2.9) can be considerably simplified by restricting the kernel $\phi(\cdot)$, which is in general a function of d (number of dimensions) scalar variables, to be a function only of the squared magnitude of its input argument:

$$\phi\left(\frac{\mathbf{x}-\mathbf{t}^{(i)}}{h_n}\right) = \phi_r\left(\left|\frac{\mathbf{x}-\mathbf{t}^{(i)}}{h_n}\right|^2\right), \tag{2.25}$$

in which $\phi_r(\cdot)$ is the restricted estimation kernel. This restriction reduces the computational requirements, but may increase the number of training samples that are necessary to yield an acceptable probability density estimate.

This special case is useful because it enables the density estimation algorithms that have been discussed in this paper to be computed using radial basis function (RBF) neural networks [Moody and Darken, 1989] [Poggio, 1990] [Specht, 1990]. Fig. 2.6 shows RBF architectures of Parzen windows, k-NN, and C-kNN density estimation. In all three architectures, the first layer of weights computes the squared distances between the input vector and each of the training vectors [as is needed for the computation of Eq. (2.25)]. For the case of Parzen windows, there is no feedback required in the network; the density estimate is simply the sum of the RBF-unit outputs. For both k-NN and C-kNN, feedback is required in order to find a value of the kernel width h (or r) such that the total output from the RBF layer is equal to the value of $k_n^{(C)}$ (or k_n) chosen *a priori*. It should be noted that all three networks shown have an important advantage compared

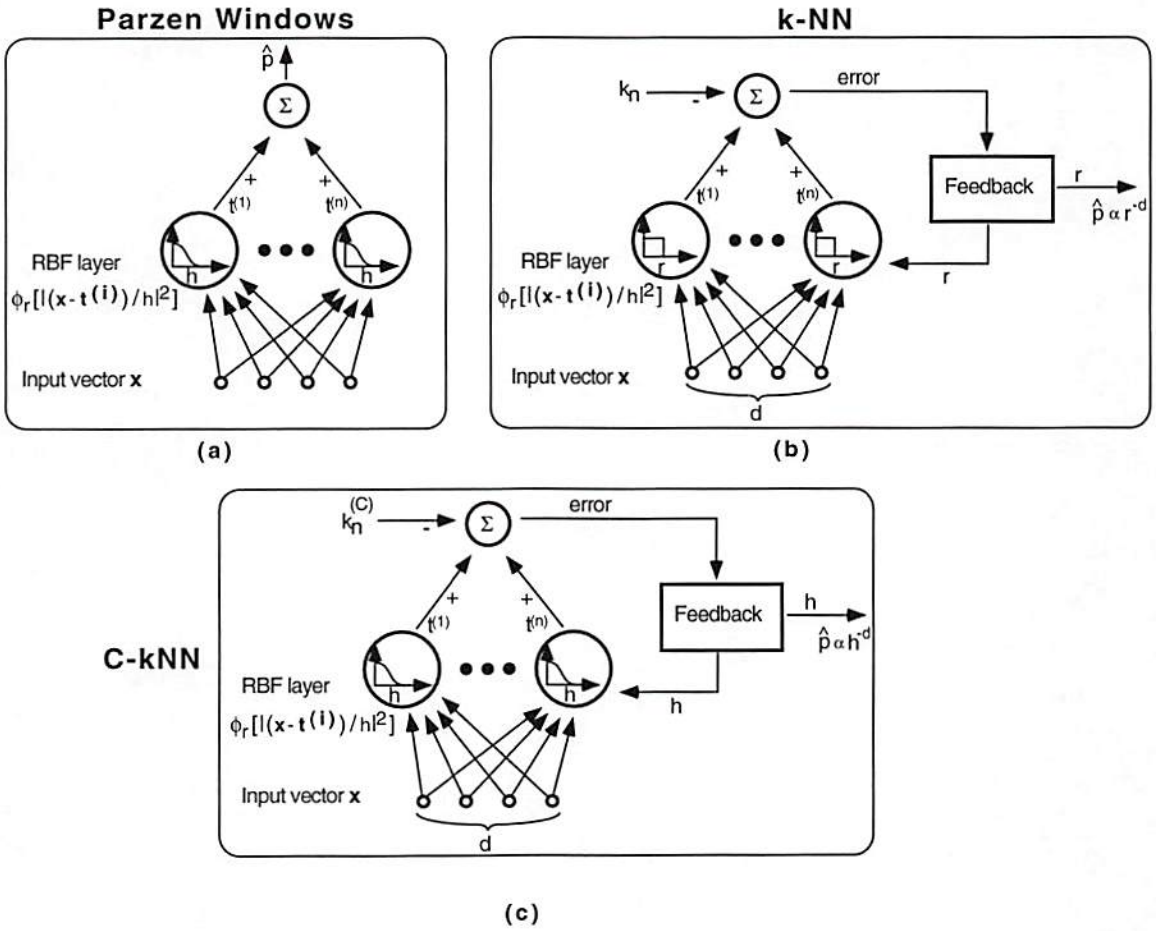


Figure 2.6: RBF architectures for (a) Parzen windows, (b) k-NN, and (c) C-kNN probability density function (PDF) estimation; each architecture is shown for the case of input vectors with dimension d=4.

to conventional serial implementations of the estimation algorithms: the neural networks compute all of the distances in parallel. This time-complexity issue has important implications for real-time systems that require density estimates, such as those discussed in Sec. 2.6.

Figure 2.6 shows an important reason why C-kNN is better suited than k-NN for analog neural network implementation: k-NN requires infinite-gain (hard threshold) neural nonlinearities, whereas C-kNN does not. When implemented on a general purpose digital computer, the hard threshold nature of k-NN is not a problem from the point of view of hardware complexity, because digital computers are very well suited for decisions such as whether or not a particular training vector should be included in the density estimate for a given input vector (*i.e.*, whether or not it is in the set of k_n nearest neighbors). However, electronic and opto-electronic hardware implementations of analog neural networks typically use differentiable (finite gain) neural units [Sheu and Choi, 1995] [Goldstein *et al.*, 1995]. In addition, it can be shown that for a given electronic hardware complexity (*i.e.*, available power, transistor characteristics, *etc.*), the amplification gain-bandwidth product is constant [Gray and Meyer, 1993]. Therefore, a neural network implementation that approaches infinite gain will necessarily be slower than one that does not have such a gain requirement. Another practical issue that favors C-kNN implementations over k-NN implementations relates to the feedback mechanism shown in Fig. 2.6. We show in Appendix C that a simple analog proportional controller [Kailath, 1980] can be used to converge to the correct kernel width $h_n(\mathbf{x})$, assuming that the kernel $\phi_r(\cdot)$ is a monotonically decreasing function of its input argument. The k-NN estimation kernel of Eq. (2.11) violates this assumption, thus requiring a more complicated search technique to find the

correct kernel width.

2.6 Applications of C-kNN: pattern classification and nonparametric regression

Once the training data has been used to estimate the underlying probability density for a given problem, the density estimate can be embedded into a larger application. The two most common applications of density estimation are pattern classification [Duda and Hart, 1973] and nonparametric regression [Hardle, 1990]. In the following, we will briefly discuss the application of C-kNN density estimation to pattern classification and nonparametric regression, as well as RBF neural network architectures for these applications.

2.6.1 Pattern classification

The fundamental problem addressed in the field of (supervised) statistical pattern classification can be stated as follows: given a set of class-labelled *i.i.d.* training vectors $\mathbf{T} = \{\mathbf{t}^{(i,c)} : i = 1 \ldots n_c, c = 1 \ldots N\}$, in which there are n_c training vectors that belong to class ω_c, classify an unknown input vector $\mathbf{x}$ (drawn from the same statistical distribution as $\mathbf{T}$) among the N possible pattern classes $\{\omega_c : c = 1 \ldots N\}$. It is well known that the statistically optimal solution to this pattern classification problem is the Bayes classifier: assign $\mathbf{x}$ to the class ω_c that maximizes the *a posteriori* probability $P(\omega_c|\mathbf{x}) \propto P(\omega_c)p(\mathbf{x}|\omega_c)$. Thus, the pattern classification problem can, in essence, be reduced to the problem of estimating a set of class-conditional probability densities, $\{p(\mathbf{x}|\omega_c)\}$. If a consistent density estimation algorithm (such as C-kNN, k-NN, or Parzen windows) is used to estimate the class-conditional probability densities, the resultant pattern classification performance is asymptotically optimal in the limit of large training

set size.

When $P(\omega_c)$ is approximated by $n_c / \sum_{j=1}^{N} n_j$, and $p(\mathbf{x}|\omega_c)$ is approximated by the C-kNN estimate given in Eq. (2.7) (where $n = n_c$), the Bayes classifier can be approximated by the following rule: assign $\mathbf{x}$ to the class ω_c such that $V_n^{(\mathrm{C})}(\mathbf{x})$ [and therefore $h_n(\mathbf{x})$] is minimized while satisfying the constraint

$$\sum_{i=1}^{n_c} \phi\left(\frac{\mathbf{x} - \mathbf{t}^{(i,c)}}{h_n(\mathbf{x})}\right) = k_n^{(\mathrm{C})}. \tag{2.26}$$

In analogy with conventional k-NN terminology [Fukunaga, 1990], we refer to the above rule as the "volumetric" C-kNN pattern classification rule. As is the case with the volumetric k-NN rule, it requires a separate determination of the kernel volume for each pattern class, and the unknown input vector is assigned to the class that yields the smallest volume.

Another formulation of the C-kNN pattern classification rule can be expressed as the following: assign $\mathbf{x}$ to the class ω_c that maximizes the expression

$$Z(\omega_c|\mathbf{x}) = \sum_{i=1}^{n_c} \phi\left(\frac{\mathbf{x} - \mathbf{t}^{(i,c)}}{h_n(\mathbf{x})}\right), \tag{2.27}$$

where $Z(\omega_c|\mathbf{x})$ is the "vote" for class ω_c, and the kernel width is chosen such that the sum of all votes is equal to a constant chosen *a priori*:

$$\sum_{c=1}^{N} Z(\omega_c|\mathbf{x}) = k_n^{(\mathrm{C})}. \tag{2.28}$$

This second rule will be referred to as the "voting" C-kNN pattern classification rule, in analogy with the "voting" k-NN rule [Fukunaga, 1990]. [Both the voting and volumetric C-kNN rules reduce to their k-NN equivalents when $\phi(\cdot)$ is chosen

as in Eq. (2.11), using similar arguments to those used previously in Sec. 2.3]. For both k-NN and C-kNN, the voting rules have the advantage that that they require only one determination of the kernel width for all classes. In Appendix D we show that, for the $N = 2$ case, the voting C-kNN rule using parameter $k_n^{(C)}$ is equivalent to the volumetric C-kNN rule using parameter $k_n^{(C)}/2$, when $\phi(\mathbf{x})$ is continuous.

As first discussed by Specht in the context of classifier design using Parzen windows density estimation, RBF neural networks map very conveniently into the probabilistic formulation of the pattern classification problem. In [Specht, 1990], Specht specifies the architectural mapping between the calculations required for Parzen windows-based classification and the weights required in a three-layered RBF neural network. Figure 2.7 (a) shows a similar neural network mapping for the case of the voting C-kNN pattern classification rule. This RBF pattern classification network operates much like the C-kNN density estimation architecture shown in Fig. 2.6, except that an additional maximum detection stage is used to determine which class has the maximum (estimated) *a posteriori* probability.

The main difference between this architecture and Specht's "Probabilistic Neural Network" (PNN) [Specht, 1990] is the presence of the C-kNN feedback mechanism that determines the kernel width as a function of the total response from the RBF layer. In contrast, the PNN uses kernel widths that are constant and independent of the input vector. The following experiment demonstrates the utility of the C-kNN feedback mechanism. A two-class pattern classification problem was simulated, in which the two classes were defined by the following Gaussian mixture densities: $p(x|\omega_1) = (1/10)\mathrm{N}(m{=}0,\ \sigma{=}100) + (9/10)\mathrm{N}(m{=}5,\ \sigma{=}1)$, and

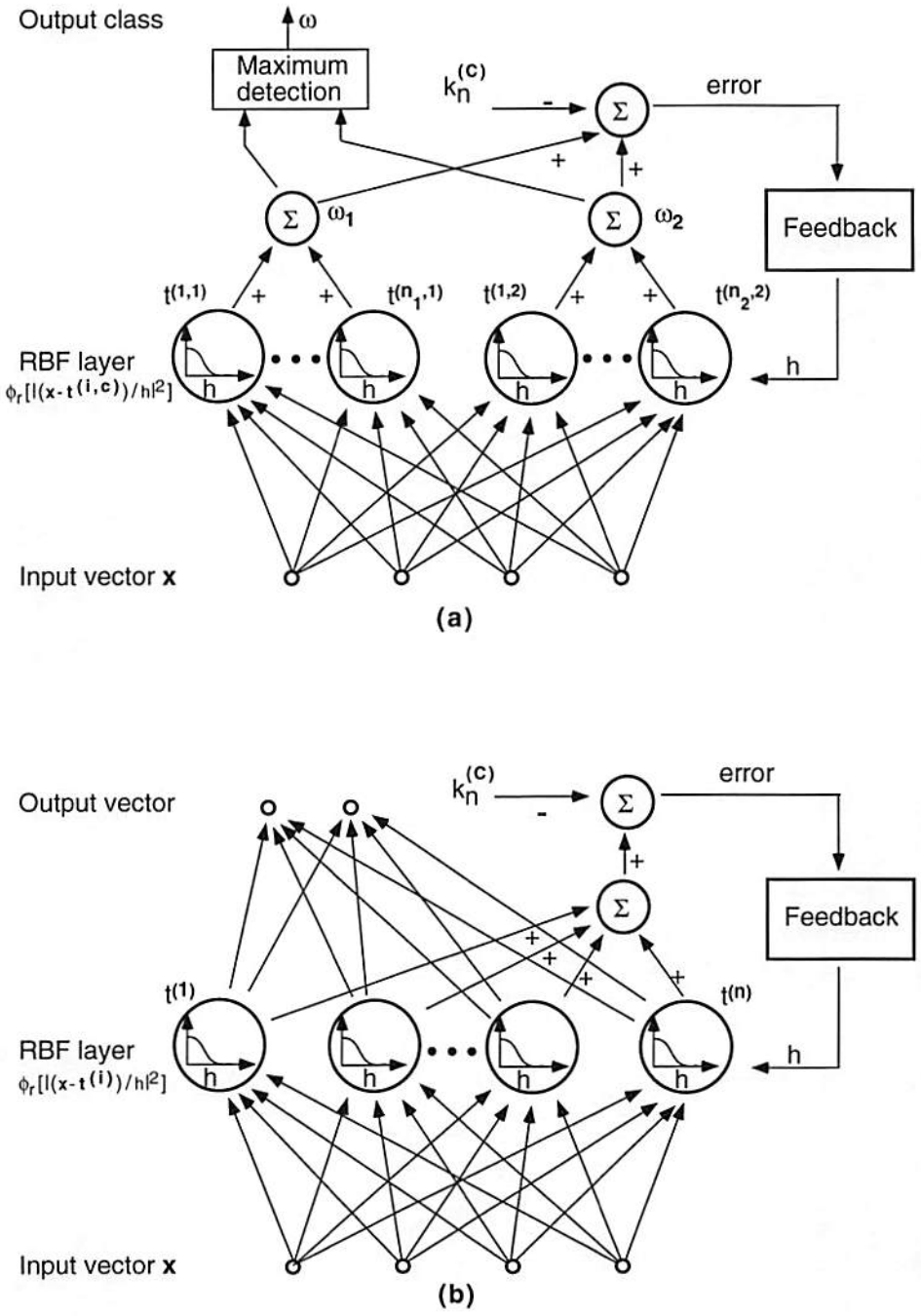


Figure 2.7: RBF architectures for C-kNN applications: (a) pattern classification (voting C-kNN rule), shown for the case of N=2 classes and input vectors of dimension 4; (b) regression, shown for the case of input vectors of dimension 4 and output vectors of dimension 2.

$p(x|\omega_2) = (1/2)\mathrm{N}(m{=}0,\ \sigma{=}100) + (1/2)\mathrm{N}(m{=}{-}5,\ \sigma{=}1)$. The *a priori* probabilities for the two classes were equal: $P(\omega_1) = P(\omega_2) = 0.5$. Such a probabilistic description is representative of applications in which two pattern classes are well separated when the input (x) is accurate, but there is a significant probability that the input has been corrupted by a very high-variance Gaussian noise source. For this classification problem, it is advantageous for the kernel width to be much smaller when $x \approx 0$ (accurate input) than when $|x| \gg 5$ (corrupted input); if the kernel width is forced to be fixed, then one would expect the performance to deteriorate for either the small x or the large $|x|$ case.

The above analysis is verified by the empirical results shown in Table 2.1. For each of the three probability density estimation methods, the (one) optimal smoothing parameter (h, k, or k^C) was determined by minimizing the estimated error probability

$$P_e = (1/T)\sum_{i=1}^{T}\left[1/n_{\mathrm{TE}}\sum_{j=1}^{n_{\mathrm{TE}}} E_{i,j}\right], \tag{2.29}$$

in which P_e denotes the estimated classification error probability, T is the number of independent trials, n_{TE} is the number of test samples for each trial, and where $E_{i,j} = 1$ if test sample j is misclassified during trial i, $E_{i,j} = 0$ otherwise. An independent set of $n = 100$ training samples and $n_{\mathrm{TE}} = 50$ test samples was generated for each trial. Averaging the results over $T = 50$ trials resulted in a percentage error $\sigma_{P_e}/P_e < 10\%$, where σ_{P_e} represents the estimated standard deviation of P_e as defined in analogy with Eq. (2.24). Gaussian kernels [$\phi(x) = \exp(-x^2/2)$] were used for both C-kNN and Parzen windows. As shown in the table, both C-kNN and k-NN significantly outperform Parzen windows for this test problem. The difference between C-kNN and k-NN ($\approx 7\%$) is too small

Table 2.1: Pattern recognition performance comparison

	Optimum parameter	Minimum error probability (P_e)
Parzen	h=0.5	0.10
k-NN	k=7	0.069
C-kNN	k^C=5.0	0.074

to be statistically significant. It should be noted that this sample problem was specifically chosen to emphasize the difference between density estimates with feedback (C-kNN, k-NN) and those without (Parzen windows); the significance of feedback would be substantially reduced for simpler classification problems.

2.6.2 Nonparametric regression

The regression problem can be formulated as follows: given n pairs of *i.i.d.* input-output training vectors $\{(\mathbf{t}^{(i)}, \mathbf{z}^{(i)}) : i = 1 \ldots n\}$, find the optimal estimate of output vector $\mathbf{z}$, denoted $\hat{\mathbf{z}}(\mathbf{x})$, when presented with a new input vector $\mathbf{x}$ drawn from the same population as $\{\mathbf{t}^{(i)}\}$. This is a statistical formulation of the standard input-output mapping problem for which supervised neural network algorithms such as backpropagation have been designed. When the criterion for optimality is the minimization of mean squared error $E\{|\mathbf{z} - \hat{\mathbf{z}}(\mathbf{x})|^2\}$, then it is well known that the statistically optimal estimate is the conditional mean, $\hat{\mathbf{z}}(\mathbf{x}) = E\{\mathbf{z}|\mathbf{x}\}$ [Papoulis, 1965]. The Parzen window probability density estimate can be used to approximate this conditional mean using the available training vector pairs,

which leads to the *Nadaraya-Watson* estimator [Hardle, 1990]

$$\hat{\mathbf{z}}(\mathbf{x}) = \frac{\sum_{i=1}^{n} \mathbf{z}^{(i)} \phi\left([\mathbf{x} - \mathbf{t}^{(i)}]/h_n\right)}{\sum_{i=1}^{n} \phi\left([\mathbf{x} - \mathbf{t}^{(i)}]/h_n\right)}. \tag{2.30}$$

Because the C-kNN algorithm adjusts the kernel width such that Eq. (2.9) holds, the denominator of Eq. (2.30) is equal to the constant $k_n^{(C)}$ for C-kNN regression:

$$\hat{\mathbf{z}}^{(C)}(\mathbf{x}) = (1/k_n^{(C)}) \sum_{i=1}^{n} \mathbf{z}^{(i)} \phi\left(\frac{\mathbf{x} - \mathbf{t}^{(i)}}{h_n(\mathbf{x})}\right), \tag{2.31}$$

where the kernel width $h_n(\mathbf{x})$ is determined by Eq. (2.9).

Specht has shown in [Specht, 1991] that the *Nadaraya-Watson* estimator maps very conveniently into an RBF neural network architecture (known as a General Regression Neural Network). Figure 2.7 (b) shows a similar architecture when C-kNN density estimation is used in place of Parzen windows. Once again, the first layer of weights computes the squared distances between the input vector and each of the training vectors. For the second layer of weights, the fan-out weight vector corresponding the the RBF center $\mathbf{t}^{(i)}$ is equal to its corresponding output vector, $\mathbf{z}^{(i)}$. There are two reasons why the C-kNN feedback mechanism can be beneficial: (1) the adaptive kernel width improves performance in some cases (as we have shown for both density estimation and pattern classification), and (2) the variable denominator of the *Nadaraya-Watson* estimator is replaced by the constant $k_n^{(C)}$ in Eq. (2.31). This second advantage is particularly important for analog hardware implementations such as that described in [Specht, 1993], in which the normalization division for the *Nadaraya-Watson* estimator must be computed off-chip. The C-kNN feedback mechanism eliminates the necessity for

such an off-chip computation, although it does require a variable-gain feedback mechanism in order to keep the denominator of Eq. (2.30) equal to a constant.

2.7 Summary and discussion

To summarize, a new probability density estimation algorithm called "Continuous-k Nearest Neighbors" has been introduced in this paper. Analytic results show that C-kNN asymptotically converges to the actual probability density from which the training samples have been drawn, and empirical simulation results have validated the analytic results. For finite-sized training sets, empirical results suggest that C-kNN has advantages over both Parzen windows and conventional k-NN density estimation: when the true density contains regions with varying spatial-frequency content (*e.g.*, Figs. 2.3-2.5), C-kNN effectively adapts the kernel widths to the local region without resulting in the jaggedness inherent to conventional k-NN estimation.

We have also shown that the C-kNN method maps straightforwardly into an analog Radial Basis Function (RBF) neural network architecture. Neural networks for C-kNN density estimation, pattern classification, and nonparametric regression were presented and compared to corresponding networks that use Parzen windows or k-NN density estimation. It was found that C-kNN outperforms Parzen windows for pattern recognition problems based upon probability densities with variable spatial-frequency content. Due to the requirement of hard threshold nonlinearities, RBF networks that use the k-NN method are not as practical for analog implementations (*e.g.*, the gain-bandwidth product and feedback issues discussed in Sec. 2.6). In contrast, both C-kNN and Parzen windows are more

easily implementable using analog electronics or opto-electronic hardware.

It should be noted that the C-kNN algorithm is much better suited to special-purpose analog hardware implementations than it is to general-purpose digital computer implementations (*i.e.*, software). In the authors' experience, the total C-kNN computation time in the software case was dominated by the feedback routines that determined the kernel width $h_n(\mathbf{x})$. We believe this difficulty was mainly due to the discrete nature of the software implementing the required feedback; oscillations in the proportional-controlled feedback loop required an adaptive mechanism (which did not always function efficiently) to monitor the results of each update to $h_n(\mathbf{x})$. A purely analog controller would not face such difficulties, as is shown analytically in App. C.

As an interesting extension of this work, it may be possible to apply the C-kNN concept to a more general class of RBF neural network architectures. In the more general context, the first layer of weights is not restricted to being a matrix consisting of the training vectors. Such a network offers a designer the flexibility of using standard clustering algorithms to determine the RBF centers [Moody and Darken, 1989], while still retaining the C-kNN feature of a feedback-controlled kernel width. Although statistical consistency theory may not underlie these more general networks, as it does for the C-kNN RBF networks discussed in this paper, the basic concept of the kernel width being dependent on the input through a feedback mechanism can still be applied.

Chapter 3

Probability density estimation for analog neural network implementation: photonic architectures

3.1 Introduction

In the previous chapter, it was noted that nonparametric probability density estimation techniques suffer from the following weaknesses when they are implemented on conventional serial computers: (1) a large number of training vectors are required for accurate estimation, which implies a large memory capacity requirement, and (2) the required computation time grows with sample size for serial implementations, thereby making real-time nonparametric pattern recognition and regression impractical. In this chapter, we present photonic probability density estimation architectures that address both of these concerns. Our basic approach is to optically implement the C-kNN and Parzen windows radial basis function (RBF) neural networks discussed in the previous chapter, using an incoherent/coherent (I/C) volume holographic architecture [Asthana *et al.*, 1993] to

store the large number of required training vectors in a compact volume. The probability density estimation architectures are then embedded into photonic architectures for both Bayes pattern classification and statistical regression applications.

3.2 Mapping between probability density estimation and optics: parallel inner products

To briefly review some definitions from the previous chapter, the Parzen window probability density estimate is defined by the equations,

$$\hat{p}_n(\mathbf{x}) = \frac{k_n(\mathbf{x})/n}{V_n}, \tag{3.1}$$

in which $\hat{p}_n(\mathbf{x})$ is the estimated probability density for the input vector $\mathbf{x}$ given n training samples $\{\mathbf{t}^{(i)} : i = 1 \ldots n\}$, and

$$k_n(\mathbf{x}) = \sum_{i=1}^{n} \phi\left(\frac{\mathbf{x} - \mathbf{t}^{(i)}}{h_n}\right). \tag{3.2}$$

The kernel volume V_n is related to the kernel width h_n through the relation $V_n = h_n^d V_0$, where d is the number of components in the input vector $\mathbf{x}$ and $V_0 = \int_{\mathbf{x}'} \phi(\mathbf{x}')d\mathbf{x}'$. For Parzen window estimation, h_n is chosen *a priori* and $k_n(\mathbf{x})$ is a random variable dependent on the input vector and training vectors. For the case of C-kNN density estimation, essentially the same equations are used, but the quantity $k_n^{(C)}$ is chosen *a priori* and the kernel width $h_n(\mathbf{x})$ is a random variable dependent on the input vector and training vectors. Thus, the photonic

implementations of Parzen windows and C-kNN estimation will be very similar, with the only difference between the two being a feedback loop similar to that discussed in Sec. (2.5) from the previous chapter.

In order to allow an efficient optical implementation of Eqs. (3.1) and (3.2), the kernel function $\phi(\cdot)$ is chosen to be a scalar function of squared radial distance, $\phi_r(\cdot)$:

$$\phi\left(\frac{\mathbf{x}-\mathbf{t}^{(i)}}{h_n}\right) = \phi_r\left(\left|\frac{\mathbf{x}-\mathbf{t}^{(i)}}{h_n}\right|^2\right) \tag{3.3}$$

$$= \phi_r\left(\frac{|\mathbf{x}|^2+|\mathbf{t}^{(i)}|^2-2\mathbf{x}^T\mathbf{t}^{(i)}}{h_n^2}\right). \tag{3.4}$$

This restriction to a radially symmetric kernel function allows an RBF neural-network implementation, as was discussed previously. In deriving a photonic implementation of an RBF neural network, the three-term expansion shown in Eq. (3.4) suggests that the fundamental operations required for probability density estimation are vector inner-products, addition, and subtraction. In the architectures proposed below, the inner-products are computed holographically, the addition is computed using optical superposition, and the subtraction is implemented using analog electronics. The division by h_n represents the variable input-scale of the estimation kernel, which will be addressed later in the discussion of the spatial-light-modulator (SLM) input gain.

Eq. (3.4) suggests a method for solving the problem of computation time increasing with training set size: if the inner-product between the input vector and all of the training vectors can be computed in parallel, then computation time becomes independent of training set size. Several planar holographic architectures

have been proposed and/or demonstrated that implement parallel inner products between an input vector and a set of stored training vectors [Caulfield, 1987] [Jang *et al.*, 1988] [Lin, 1990] [Lu and Lin, 1991] [Goldstein and Jenkins, 1992]. These systems use holographic film as the recording medium, which has previously been extensively characterized in the literature with respect to its multiplexing capabilities (*e.g.*, [Shamir *et al.*, 1989] [Johnson *et al.*, 1984]). Optical implementations of inner products using optical disks and/or incoherent matrix-vector multipliers have also been previously discussed in the context of squared-distance calculations [Neifeld and Psaltis, 1993]. However, a fundamental limitation inherent to any 2-D architecture is that the maximum recordable spatial frequency is limited by the wavelength of the recording light. That is, the maximum number of resolution elements is limited to the total available area divided by λ^2. Volume holographic media are theoretically capable of much higher storage capacity because they are fundamentally limited to the total available *volume* divided by λ^3 [van Heerden, 1963].

3.2.1 Incoherent/Coherent volume holography

The incoherent/coherent double angularly multiplexed (I/C) architecture [Jenkins *et al.*, 1990] [Asthana *et al.*, 1993] can be employed to store the training vectors in a volume holographic element (VHOE), as shown in Fig. 3.1. In this "sub-hologram" version of the architecture, the holograms are recorded using both spatial and angular multiplexing, which yields high optical throughput, low crosstalk, and minimizes material dynamic-range requirements [Asthana *et al.*, 1993].

During holographic recording, an input-SLM reference beam R_j, in which j ranges from $[1 \ldots d]$, interferes with a training-SLM object beam consisting of a set

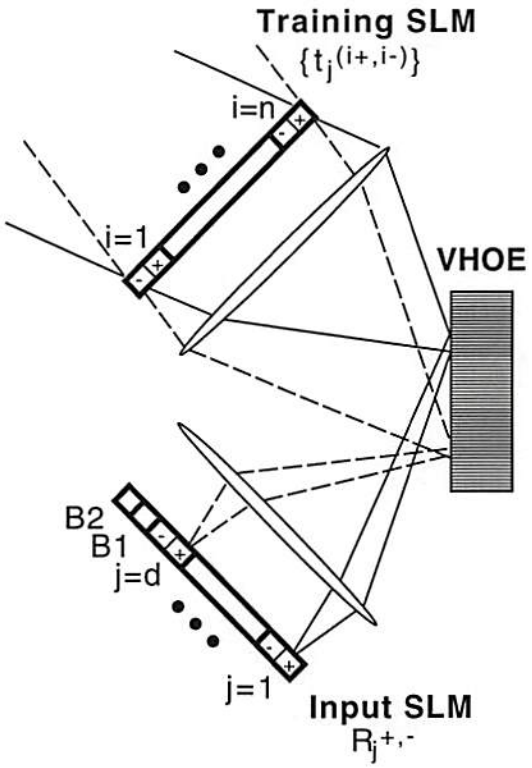


Figure 3.1: Recording dual-rail training vectors

of pixels that intensity-encodes component j from the set of training vectors. This set of pixels is denoted $\{t_j^{(i)} : i = 1 \ldots n\}$, in which n is the number of training vectors. Each training-vector component set $\{t_j^{(i)} : i = 1 \ldots n\}$ is sequentially placed at the training SLM and is recorded with a single reference R_j. The training SLM is illuminated by an angularly-multiplexed plane-wave for each holographic recording, resulting in d partially overlapping sub-holograms. The degree to which the sub-holograms overlap depends on the specifics of the angular multiplexing and placement of the lenses. It should be noted that all SLM's shown in the figures in this chapter are assumed to be physically two dimensional; they are drawn as one-dimensional objects for simplicity. Similarly, the training "vectors" we refer to in this chapter are optically represented as two-dimensional objects.

Because the incoherent/coherent architecture is intensity-encoded, a dual-rail representation must be used in order to accommodate bipolar training and input vectors. At the training SLM, this dual-rail encoding can be accomplished by decomposing each bipolar quantity $t_j^{(i)}$ into two positive quantities $t_j^{(i+)}$ and $t_j^{(i-)}$, such that $t_j^{(i)} = t_j^{(i+)} - t_j^{(i-)}$. Each training vector component $t_j^{(i)}$ is optically represented at the training SLM with two adjacent pixel intensities, $I_{wo}t_j^{(i+)}$ and $I_{wo}t_j^{(i-)}$, in which I_{wo} is the object beam writing intensity (we assume that $t_j^{(i+)}$ and $t_j^{(i-)}$ have been normalized to the maximum SLM reflectivity). Each input SLM reference R_j is also dual-rail encoded, in the following sense. Every training-vector component set is recorded twice (sequentially): the set $\{t_j^{(i)}\}$ is recorded with reference R_j^-, and its negation $\{-t_j^{(i)}\}$ (*i.e.*, adjacent + and − components are swapped) is recorded with reference R_j^+. The reference beams R_j^+ and R_j^- have the same intensity, I_{wr}. By dual-rail encoding the reference beams during recording, we allow the use of bipolar input vectors during readout.

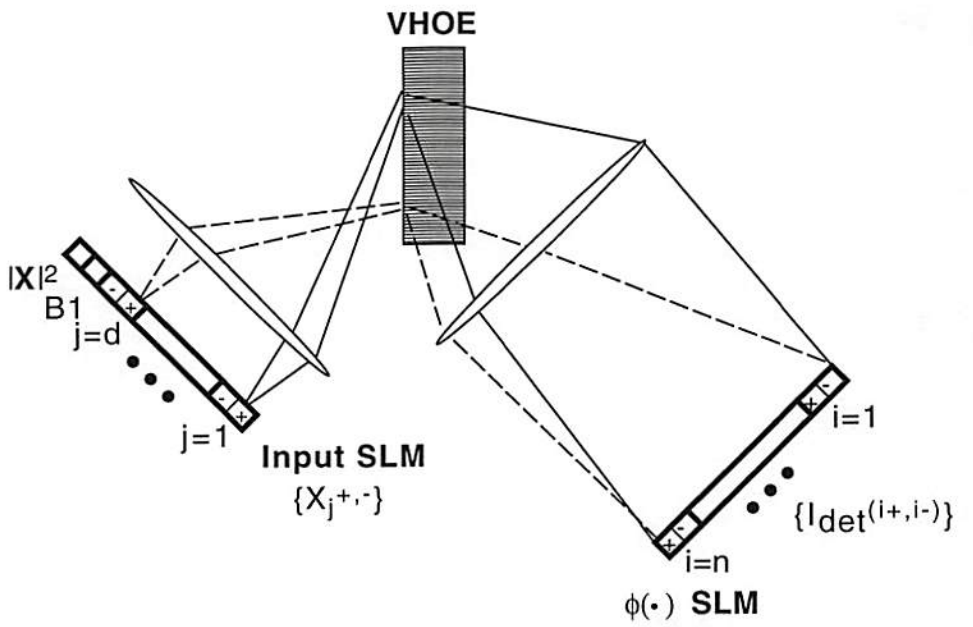


Figure 3.2: Reading out dual-rail training vectors

Two additional sub-holograms, besides the $2d$ sub-holograms discussed above, must be recorded in order to compute Eq. (3.4): one computes the $|\mathbf{t}^{(i)}|^2$ term and the other computes the $|\mathbf{x}|^2$ term. The first of these sub-holograms is stored by setting the $2n$ training-SLM pixel intensities according to $t^{(i+)} = (I_{wo}/d)\sum_{j=1}^{d}(t_j^{(i)})^2$, $t^{(i-)} = 0$ and recording with reference beam B1 (intensity=I_{wr}). The last term to be computed, $|\mathbf{x}|^2$, is not a function of training vector index i. Thus, a uniform hologram $t^{(i+)} = I_{wo}$, $t^{(i-)} = 0$ is recorded with reference B2 (intensity = I_{wr}).

During holographic readout, the input SLM is read out by a set of individually coherent but mutually-incoherent point sources [Asthana *et al.*, 1993]. The SLM is addressed by a dual-rail representation of the input vector $\mathbf{x} = \{x_j\}$, such that $x_j^+ - x_j^- = x_j$ (see Fig. 3.2). Each pixel is read out with intensity $(2I_r)x_j^{+,-}$, in which $2I_r$ is an SLM pixel-readout intensity (we assume that x_j^+ and x_j^- have been normalized to the maximum SLM reflectivity). Input SLM pixel B1 is read

out with the constant intensity, $(d \cdot I_r)$. Input SLM pixel B2 is read out with an intensity proportional to an externally computed quantity, $(d \cdot I_r)[1/d\sum_{j=1}^{d}(x_j)^2]$, where the summation is computed either optically or electronically with additional hardware. The $\phi(\cdot)$ SLM detectors are placed in the image plane of the training SLM, resulting in a weighted incoherent superposition of the angularly multiplexed training data. The difference between the two detected intensities, $I_{det}^{(i+)} - I_{det}^{(i-)}$, is electronically computed at each $\phi(\cdot)$ SLM pixel, as follows:

$$\begin{aligned}
I_{det}^{(i+)} &= (\eta I_r)\left[2\sum_{j=1}^{d}\left(x_j^- t_j^{(i+)} + x_j^+ t_j^{(i-)}\right) + |\mathbf{x}|^2 + |\mathbf{t}^{(i)}|^2\right] \\
I_{det}^{(i-)} &= (\eta I_r)\left[2\sum_{j=1}^{d} x_j^- t_j^{(i-)} + x_j^+ t_j^{(i+)}\right] \quad \Longrightarrow \\
I_{det}^{(i+)} - I_{det}^{(i-)} &= (\eta I_r)\left[\sum_{j=1}^{d}\left(-2x_j t_j^{(i)}\right) + |\mathbf{x}|^2 + |\mathbf{t}^{(i)}|^2\right] \\
I_{det}^{(i+)} - I_{det}^{(i-)} &= (\eta I_r)\left(|\mathbf{x}|^2 + |\mathbf{t}^{(i)}|^2 - 2\mathbf{x}^T\mathbf{t}^{(i)}\right), \qquad (3.5)
\end{aligned}$$

in which $\eta < 1$ is a loss factor dependent on the VHOE material, the two writing intensities I_{wo} and I_{wr}, and the recording exposure time. (For simplicity, we have assumed that the modulators and detectors have equal areas.) Thus, the distance-squared calculation required for an RBF neural-network implementation can be optically computed using the I/C sub-hologram architecture and a dual-rail SLM.

3.2.2 Variable gain SLM

We described above the method by which the set of squared distances $\{|\mathbf{x}-\mathbf{t}^{(i)}|^2\}$ can be computed in parallel. The last step that remains in order to compute Eq. (3.4) is the incorporation of the variable width h_n and kernel function $\phi_r(\cdot)$.

The kernel width h_n can be adjusted by varying the SLM input gain (see Fig. 3.3). The figure shows a system-level model of a spatial light modulator, as defined by

$$I_{out}^{(i)} = \phi_{SLM}\left[G_n(I_{det}^{(i+)} - I_{det}^{(i-)})\right], \tag{3.6}$$

in which $I_{out}^{(i)}$ is the SLM output intensity, $\phi_{SLM}(\cdot)$ is the SLM transfer function, $I_{det}^{(i+)}$ and $I_{det}^{(i-)}$ are the dual-rail detected intensities, and G_n is the variable electronic input gain. Each SLM pixel functions as follows. The input intensities are transduced into dual-rail voltages that are electronically differenced with amplification G_n. The resultant voltage V_{in} modulates the reflectivity (or transmissivity) of a light modulator R_{out}, and the modulator is read out with an external source yielding the output intensity $I_{out}^{(i)}$. At the system level, it is sufficient to model these multiple opto-electronic transductions as an optical-input/optical-output transfer function given by Eq. (3.6).

By relating SLM gain to kernel width as

$$h_n = 1/\sqrt{G_n \eta}, \tag{3.7}$$

and substituting Eq. (3.7) into Eq. (3.5) combined with Eq. (3.6), we find that

$$I_{out}^{(i)} = \phi_{SLM}\left(I_r \left|\frac{\mathbf{x} - \mathbf{t}^{(i)}}{h_n}\right|^2\right). \tag{3.8}$$

If the density estimation kernel function $\phi(\cdot)$ is related to the SLM transfer function $\phi_{SLM}(\cdot)$ through the relation

$$\phi(p) = (1/I_r)\phi_{SLM}(pI_r), \tag{3.9}$$

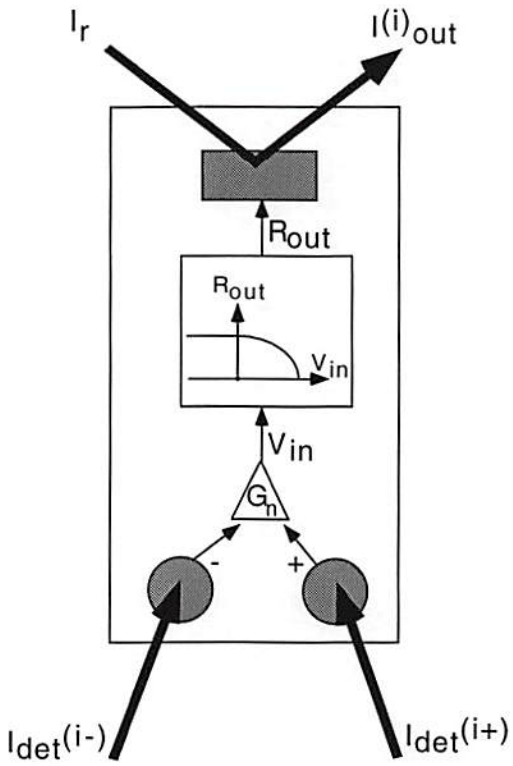


Figure 3.3: System-level SLM model

where p is a dummy variable, then

$$I_{out}^{(i)} = I_r \phi \left(\left| \frac{\mathbf{x} - \mathbf{t}^{(i)}}{h_n} \right|^2 \right). \tag{3.10}$$

Thus, the modulated output from each SLM pixel yields a scaled version of the required kernel function $\phi(\cdot)$ in Eq. (3.4). By altering the SLM gain G_n, the kernel width h_n changes in accordance with Eq. (3.7).

3.3 Photonic architectures for C-kNN and Parzen window probability density estimation

In the following two subsections, photonic volume holographic architectures for probability density estimation are described. The Parzen window and C-kNN consistency conditions are related to the SLM gain in order to ensure statistical consistency in the photonic architectures.

3.3.1 Parzen window estimation: SLM gain scheduling

As discussed previously, Parzen window probability density estimation is statistically consistent if the kernel volume satisfies the conditions $V_n \to 0$ and $nV_n \to \infty$, as $n \to \infty$. These two conditions, together with Eq. (3.7), lead to the SLM *gain scheduling* conditions,

$$G_n \to \infty \tag{3.11}$$

$$G_n^{d/2}/n \to 0. \tag{3.12}$$

Fig. 3.4 is a schematic of a photonic Parzen window probability density estimation system, in which the training vectors $\{\mathbf{t}^{(i)}\}$ have been stored according to Sec. 3.2.1, the input vector $\mathbf{x}$ is represented in dual-rail form at the input SLM, and the SLM gain schedule conditions Eqs. (3.11) and (3.12) are assumed to be satisfied at the $\phi(\cdot)$ SLM. The area-detected intensity $I_n(\mathbf{x})$ is proportional to the Parzen window estimate $\hat{p}_n(\mathbf{x})$, as shown by the following: the area detector integrates the total output light reflected off of the $\phi(\cdot)$ SLM, yielding

$$I_n(\mathbf{x}) = \sum_{i=1}^{n} I_{out}^{(i)} = I_r \sum_{i=1}^{n} \phi\left(\left|\frac{\mathbf{x}-\mathbf{t}^{(i)}}{h_n}\right|^2\right) = I_r k_n(\mathbf{x}), \tag{3.13}$$

using Eq. (3.10) and Eq. (3.2), and therefore

$$\hat{p}_n(\mathbf{x}) = \frac{I_n(\mathbf{x})/I_r}{nV_n} \tag{3.14}$$

using Eq. (3.1).

3.3.2 C-kNN density estimation: feedback-controlled SLM gain

We showed earlier that the C-kNN probability density estimation is statistically consistent if the conditions $k_n^{(C)} \to \infty$ and $k_n^{(C)}/n \to 0$ as $n \to \infty$. The detected intensity I_n is proportional to $k_n^{(C)}$, as shown by Eq. (3.13), implying that the application of these conditions to the *detection schedule* I_n will guarantee statistical consistency:

$$I_n \;\to\; \infty \tag{3.15}$$

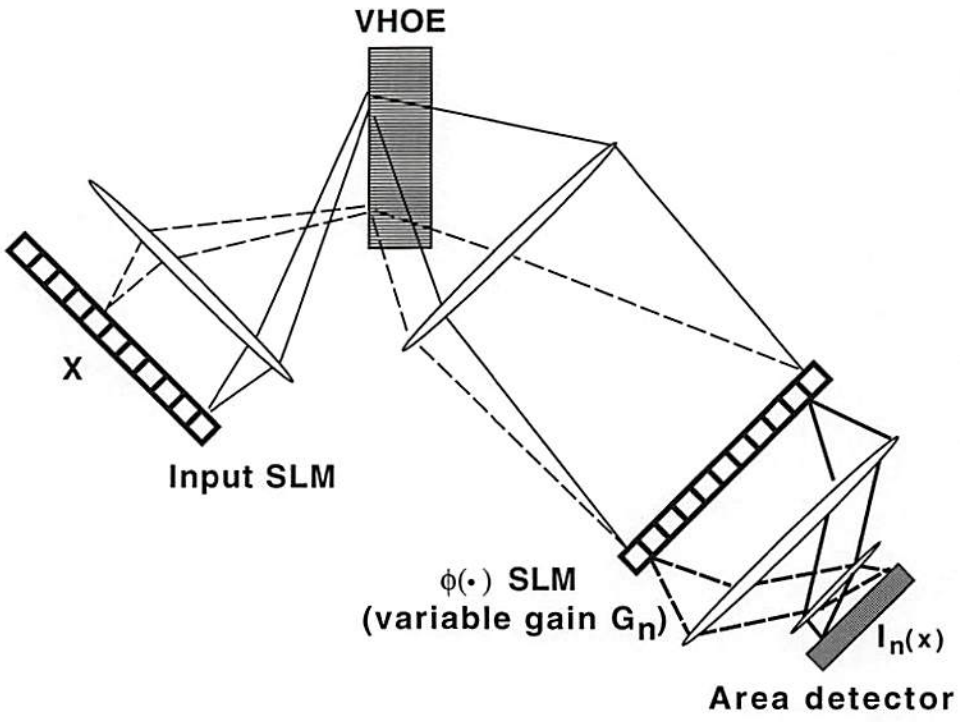


Figure 3.4: Photonic architecture for probability density estimation. The area detector integrates the total output light reflected off the $\phi(\cdot)$ SLM. For Parzen window estimation, the estimated probability density is proportional to the area-detected intensity. For C-kNN estimation, the estimated density is a function of the feedback-determined SLM gain required to detect an intensity chosen *a priori*.

$$I_n/n \rightarrow 0. \tag{3.16}$$

The $\phi(\cdot)$ SLM gain $G_n(\mathbf{x})$ is determined for a given input vector $\mathbf{x}$ by forcing the total SLM output intensity to be equal to the scheduled value I_n, through the use of a feedback loop that alters the SLM gain until the detected intensity equals the value chosen *a priori*. By combining Eq. (3.7) with the form of Eqs. (3.1) and (3.2) appropriate to C-kNN, it can be shown that the C-kNN probability density estimate is given by

$$\hat{p}_n^{(C)}(\mathbf{x}) = [G_n(\mathbf{x})^{d/2}] \left(\frac{\eta^{d/2} k_n^{(C)}}{n V_0} \right), \tag{3.17}$$

which is a function of the feedback-determined SLM gain.

3.4 Photonic architectures for Bayes pattern classification and nonparametric regression applications

The probability density architectures discussed in the previous section can now be embedded into larger systems that use density estimates. In the first subsection, a photonic architecture that implements an approximation to a Bayes classifier is described. The optical system uses either Parzen windows or C-kNN density estimation to approximate the *a posteriori* probability that the input vector belongs to a given pattern class. In the second subsection, a photonic nonparametric regression architecture is presented, which uses either Parzen windows or C-kNN density estimation to approximate a minimum mean squared error mapping between input and output vectors.

3.4.1 Photonic Bayes classifier

To briefly review some definitions from the previous chapter, the statistical pattern classification task can be stated as follows: given a set of training vectors $\{\mathbf{t}^{(i,c)} : i = 1 \ldots n_c,\ c = 1 \ldots N\}$ in which there are n_c training vectors $\{\mathbf{t}^{(i,c)}\}$ that belong to class ω_c, classify an unknown input vector $\mathbf{x}$ (drawn from the same statistical distribution as the training vectors) among the N possible pattern classes $\{\omega_c : c = 1 \ldots N\}$. The Bayes classifier gives the statistically optimal solution: assign $\mathbf{x}$ to the class ω_c that maximizes the *a posteriori* probability $P(\omega_c|\mathbf{x}) \propto P(\omega_c)p(\mathbf{x}|\omega_c)$. If $P(\omega_c)$ is approximated by $n_c/\sum_{j=1}^{N} n_j$ and $p(\mathbf{x}|\omega_c)$ is approximated by the Parzen window estimate from Eq. (3.1), then the Bayes classifier can be approximated by the following rule: assign $\mathbf{x}$ to the class that maximizes the expression

$$Z(\omega_c|\mathbf{x}) = \sum_{i=1}^{n_c} \phi\left(\frac{\mathbf{x} - \mathbf{t}^{(i,c)}}{h_n}\right), \tag{3.18}$$

where $Z(\omega_c|\mathbf{x})$ is the "vote" for class ω_c, and h_n is determined *a priori*. For the case of C-kNN estimation the same rule applies, except that the kernel width $h_n(\mathbf{x})$ is determined by enforcing the following condition:

$$\sum_{c=1}^{N} Z(\omega_c|\mathbf{x}) = k_n^{(C)}, \tag{3.19}$$

in which $k_n^{(C)}$ is chosen *a priori*.

When Bayes pattern classifiers are implemented using serial computers, it is

difficult to obtain real-time performance because the probability density estimation time grows with training set size. The photonic probability density estimation architectures discussed here solve this problem because all of the required inner-product computations are done in parallel. In addition, current multiple quantum well SLM's operate in the MHz range, which is more than sufficient for real-time image processing applications that typically require a 30 Hz clock rate (30 frames/sec.).

Eqs. (3.18) and (3.19) have straightforward optical implementations given the results derived in Sec. 3.3. For pattern recognition applications, during holographic recording (Fig. 3.1) the training vector pixels are spatially separated at the training SLM according to class membership. (*i.e.*, a distinct spatial region is devoted to each class ω_c). Thus, during holographic readout the votes $Z(\omega_c|\mathbf{x})$ can be computed by area-detecting each class separately through the use of a microlens array, as shown in Fig. 3.5. Using an expression similar to Eq. (3.13), it can be shown that the area-detected intensity $I(\omega_c|\mathbf{x})$ for class ω_c is given by

$$I(\omega_c|\mathbf{x}) = I_r Z(\omega_c|\mathbf{x}), \tag{3.20}$$

which implies that a maximum detection stage following the area detectors yields the Parzen window estimate of a Bayes classification. Similarly, the C-kNN estimate of a Bayes classification is obtained by adjusting the $\phi(\cdot)$ SLM gain $G_n(\mathbf{x})$ until the total intensity $\sum_{i=1}^{c} I(\omega_c|\mathbf{x}) = I_r k_n^{(C)}$, and once again using a maximum detection stage for the classification decision.

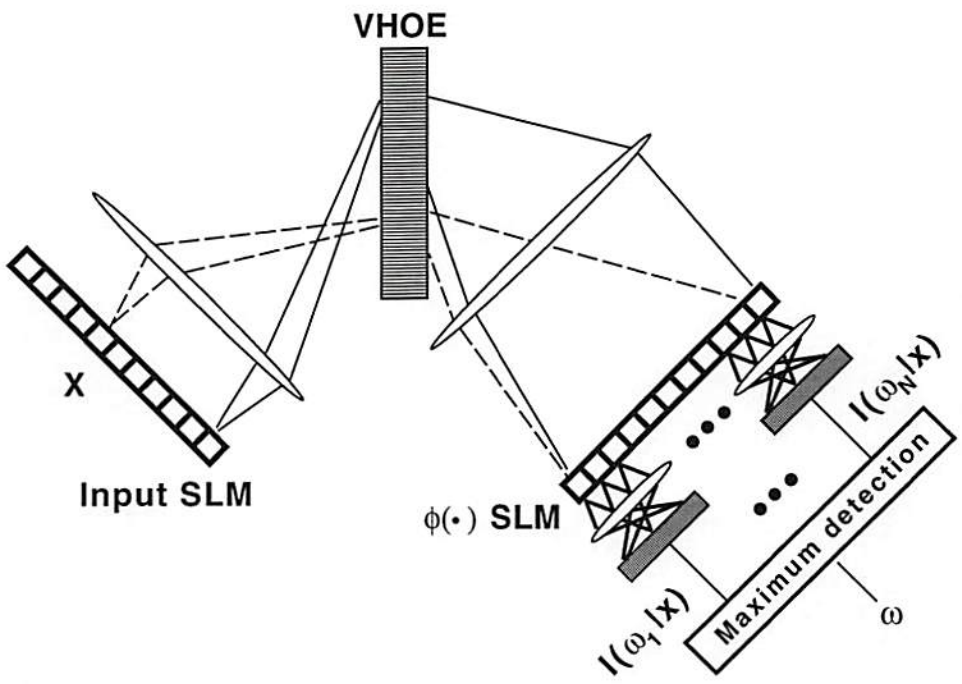


Figure 3.5: Photonic architecture for a Bayes classifier

3.4.2 Photonic nonparametric regression

As discussed in the previous chapter, another common application of probability density estimation is nonparametric regression [Hardle, 1990]. To briefly review, the regression problem can be formulated as follows: given n pairs if $i.i.d$ input-output training vectors $\{(\mathbf{t}^{(i)}, \mathbf{z}^{(i)}) : i = 1 \ldots n\}$, find the best estimate of $\mathbf{z}$, denoted $\hat{\mathbf{z}}(\mathbf{x})$, when presented with a new input vector $\mathbf{x}$ (assuming $\mathbf{x}$ is drawn from the same distribution as $\{\mathbf{t}^{(i)}\}$). If the "best" estimate is defined as the one with minimum mean squared error (MMSE) $E\{|\mathbf{z} - \hat{\mathbf{z}}(\mathbf{x})|^2\}$, then the optimal estimate is known to be the conditional mean $\hat{\mathbf{z}}(\mathbf{x}) = E\{\mathbf{z}|\mathbf{x}\}$. The Parzen window probability density estimate can be used to approximate this conditional mean,

leading to the so-called *Nadaraya-Watson estimator* [Hardle, 1990]

$$\hat{\mathbf{z}}(\mathbf{x}) = \frac{\sum_{i=1}^{n} \mathbf{z}^{(i)} \phi\left(\frac{\mathbf{x}-\mathbf{t}^{(i)}}{h_n}\right)}{\sum_{i=1}^{n} \phi\left(\frac{\mathbf{x}-\mathbf{t}^{(i)}}{h_n}\right)}, \tag{3.21}$$

in which h_n is a deterministic function of n satisfying the Parzen convergence conditions. If the C-kNN estimate is used instead of Parzen windows, a similar conditional mean is obtained,

$$\hat{\mathbf{z}}^{(C)}(\mathbf{x}) = \frac{\sum_{i=1}^{n} \mathbf{z}^{(i)} \phi\left(\frac{\mathbf{x}-\mathbf{t}^{(i)}}{h_n(\mathbf{x})}\right)}{k_n}, \tag{3.22}$$

where the kernel width $h_n(\mathbf{x})$ is determined through feedback by forcing the denominator of Eq. (3.21) to be equal to the predetermined value k_n.

A photonic implementation of nonparametric regression is (not surprisingly) very similar to photonic implementations of probability density estimation and optical Bayes classification. However, two VHOE's are required in this case: the first VHOE stores the set of input training vectors $\{\mathbf{t}^{(i)}\}$ and the second VHOE stores the set of output training vectors $\{\mathbf{z}^{(i)}\}$. The input training vectors are recorded using the same method discussed previously for probability density estimation (Fig. 3.1). The output training vectors, however, are recorded somewhat differently. The first difference is that Eqs. (3.21) and (3.22) require a weighted superposition of the $\{\mathbf{z}^{(i)}\}$ vectors, which implies that the pixels B1 and B2 in Fig. 3.1 (used for a squared-distance calculation) are not needed. Secondly, the output vectors are stored using the "transpose" of the previous recording method. That is, each output vector $\{z_j^{(i)} : j = 1 \ldots d\}$ is sequentially recorded as a subhologram in the second VHOE (n recordings of d SLM pixels), whereas each

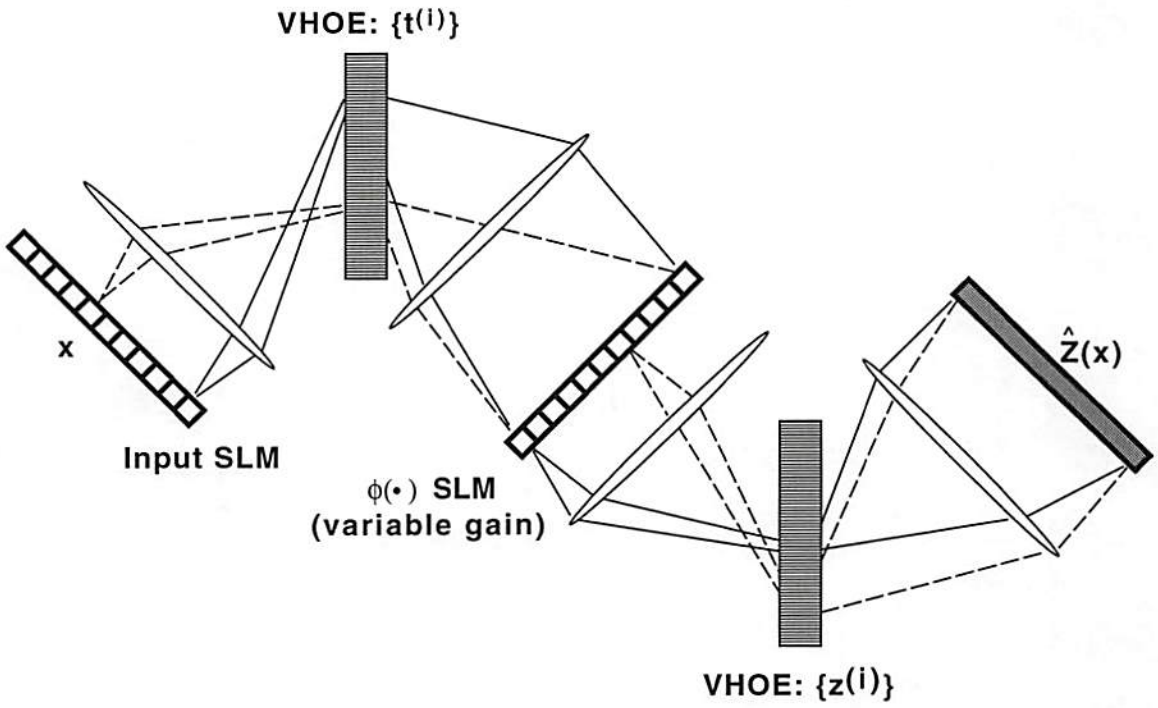


Figure 3.6: Photonic architecture for nonparametric regression

training-vector component set $\{t_j^{(i)} : i = 1 \ldots n\}$ is recorded as a subhologram in the first VHOE (d recordings of n SLM pixels).

Fig. 3.6 shows the proposed optical regression architecture. As was the case in the previous discussion, both Parzen window and C-kNN regression use the same optics, but for C-kNN estimation the $\phi(\cdot)$ SLM gain is determined by forcing the total output intensity of $\phi(\cdot)$ SLM to be a pre-determined value dependent on n. One important difference between the two, however, is that the C-kNN version does not need external electronic hardware to compute the division in Eq. (3.21), because the denominator is independent of the input vector $\mathbf{x}$ [Eq. (3.22)].

3.5 An alternate holographic recording method

One possible disadvantage in the photonic architectures described in this chapter is that our method for holographically storing the training vectors, given in Sec. (3.2.1), requires the use of a digital computer to initially store all of the vectors before they can be holographically recorded. This digital storage is necessary because a training-vector component set $\{t_j^{(i)} : i = 1 \ldots n\}$ can be placed at the training SLM only after all of the training vectors $\{\mathbf{t}^{(i)} : i = 1 \ldots n\}$ have been collected. Furthermore, if any additional training vectors are subsequently obtained, all of the original d sub-holograms need to be rewritten; incremental addition of new training data is not possible.

To resolve the two issues raised above, we propose an alternate recording scheme briefly described below. The alternate recording scheme has the advantage that the training vectors can be directly stored in the hologram, without first collecting and digitally storing all n vectors. However, the alternative method also has the significant disadvantage of being limited by a planar-holographic storage capacity (as will be explained below). Thus, if all training vectors are simultaneously available and can be digitally stored, the original volume holographic recording architecture is preferable.

The two-step recording scheme shown in Fig. 3.7 shows one method for directly storing the training vectors [Goldstein and Jenkins, 1992]. (A previously reported holographic recording technique [Caulfield, 1987] also allows direct storage, but this technique may not be practical due to requirements on the SLM [Shamir *et al.*, 1989]) As shown in the figure, during the first recording step each training vector (actually a 2-D image) is placed at the training SLM, and a set of

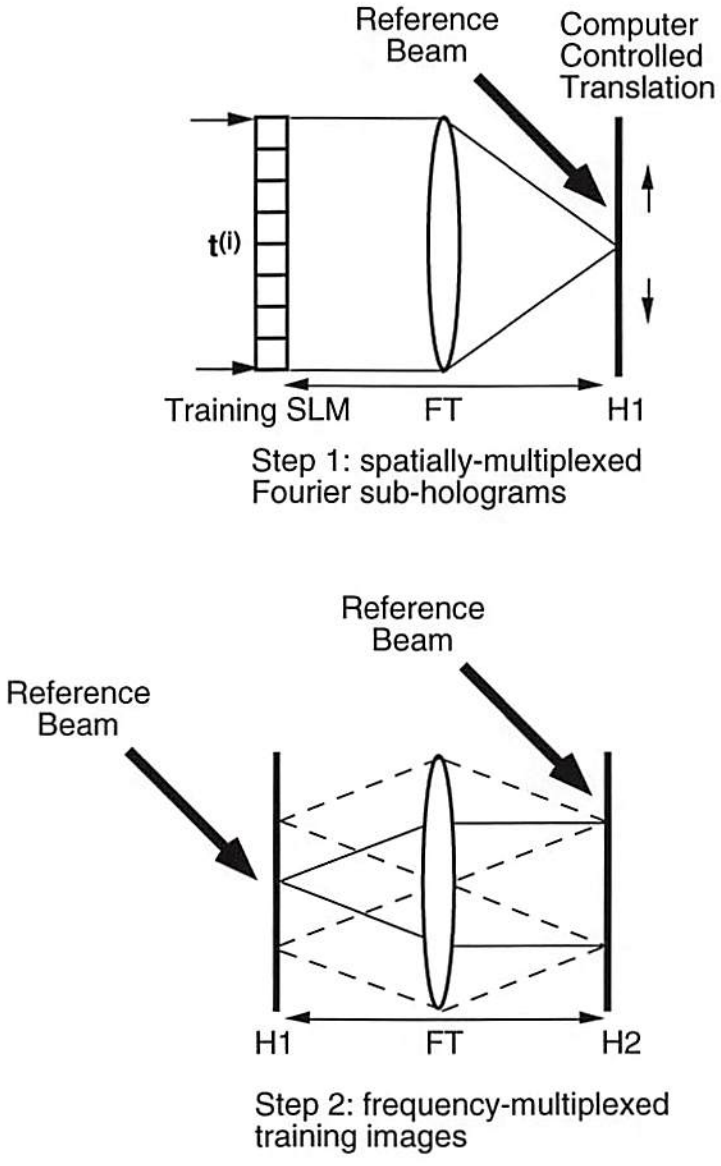


Figure 3.7: Two-step holographic recording process

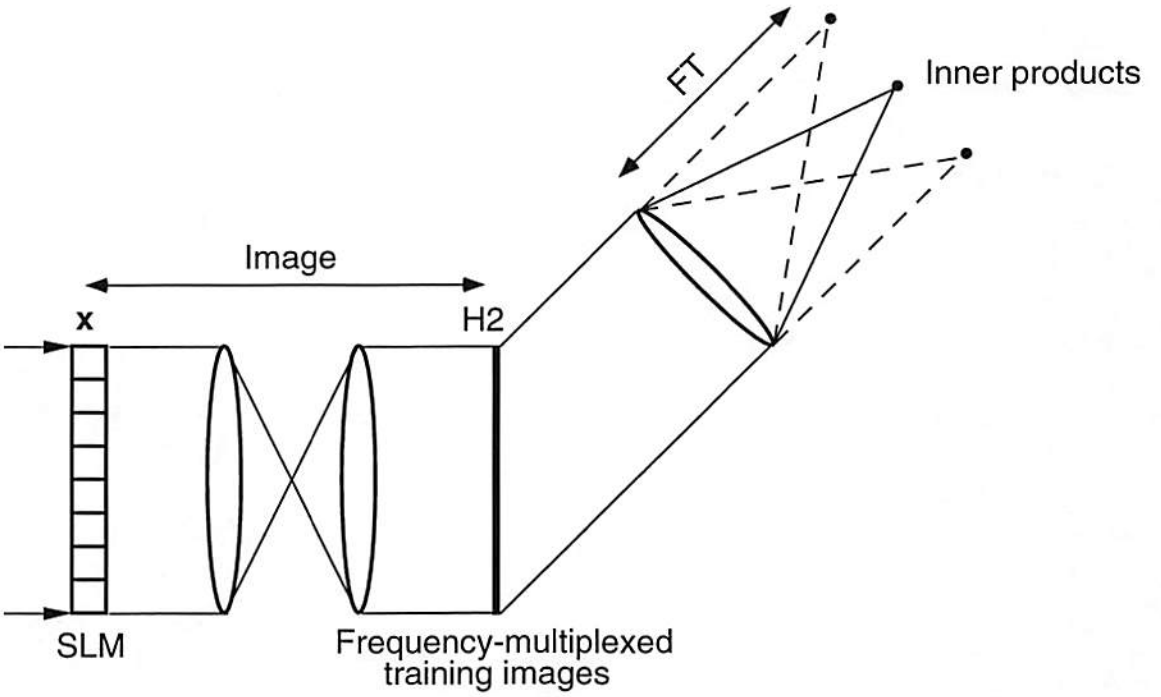


Figure 3.8: Holographic readout: parallel inner products between the input vector and all training vectors.

spatially non-overlapping Fourier sub-holograms is recorded in hologram H1 using a computer-controlled translation stage. In order to accommodate bipolar training and input vectors, as well as the three-term distance-squared calculation required for probability density estimation, the same training-SLM optical representations that were discussed in Sec. 3.2.1 can be employed. During the second recording step, each of the spatially separated Fourier sub-holograms is converted back into the originally recorded image, through the use of a second (global) Fourier-transforming lens. Each of these superimposed images will be recorded with a different spatial-frequency carrier at hologram H2, since the images approaching the second hologram are angularly-multiplexed [Goodman, 1996]. Thus, hologram H2 consists of a set of spatial-frequency multiplexed images. If new training vectors become available at a later time, it may be possible to record additional

holograms on hologram H1 (without rewriting all the other holograms), but H2 would need to be rewritten.

When an input vector is imaged (in both amplitude and phase) onto the spatial-frequency multiplexed hologram H2 (see Fig. 3.8), a point-wise multiplication takes place between the input vector and each of the training vectors. A Fourier-transforming lens collects the first-order diffracted light from H2, forming a set of spots in the focal plane in which the integrated intensity of a given spot is proportional to the inner product between the input vector and a corresponding training vector. In order for the inner-product spots to be spatially separated at the focal plane, the training vectors stored in H2 must have non-overlapping 2-D spatial-frequency content. Herein lies the storage capacity limitation mentioned previously: because the stored images must be non-overlapping in 2-D spatial-frequency content, the total number of pixels that can be stored is limited by the 2-D space-bandwidth product of H2 [Goodman, 1996]. Thus, even if H2 is a volume hologram, the storage capacity is inherently limited to the cross-sectional area divided by λ^2.

3.6 Discussion

This chapter has addressed some of the issues involved in the photonic implementation of pattern recognition and regression systems that use nonparametric probability density estimation techniques. These photonic architectures are based on the radial basis function (RBF) neural networks discussed in the previous chapter. A volume holographic architecture that computes the required squared-distance calculations in parallel, based on the sub-hologram version of the incoherent/coherent architecture [Asthana *et al.*, 1993], was described in detail.

Photonic architectures were presented that implement nonparametric probability density estimation, real-time Bayes classification, and real-time nonparametric regression. The architectures described were applied to both C-kNN and Parzen window density estimation.

An important issue we have not addressed in this chapter is that the number of training vectors needed for acceptable system performance may exceed the storage capacity of even a volume hologram. We have shown that the photonic systems will converge to statistical optimality in the limit of an infinite number of training vectors, but in practice an infinite number of samples cannot be stored in a finite medium. Although this is clearly an important issue, nonparametric techniques such as k-NN and Parzen windows have proven to be useful in practice [Geman *et al.*, 1992] [Specht, 1990] [Specht, 1991] [Hardle, 1990] [Silverman, 1986] [Dasarathy, 1991] with only a finite amount of training data. One particularly troubling weakness inherent to nonparametric estimation techniques is the so-called "curse of dimensionality" [Bellman, 1961], which states that the number of training vectors required increases exponentially with the number of dimensions in each training vector. However, it has been shown that when the surface being estimated is sufficiently smooth the curse of dimensionality can be avoided [Silverman, 1986] [Xu *et al.*, 1994]. Whether or not this smoothness requirement is met in typical applications is unknown, but the fact that nonparametric techniques have proven themselves useful suggests that the curse of dimensionality can be avoided when systems are properly designed.

Chapter 4

Gain and exposure scheduling to compensate for photorefractive neural-network weight decay

4.1 Introduction

In the previous two chapters, we applied statistical techniques to the design of optical neural-network systems that learn a relationship between input and output vectors given training set data. These systems work by holographically storing the set of training vectors; when the system is presented with a new input, it interpolates between the stored vectors in a statistically optimal sense. These statistical techniques have a number of advantages over other proposed adaptive neural network pattern recognition systems: nonlinear classification is allowed (as opposed to optical synthetic discriminant function methods [Casasent, 1984] and Perceptron classifiers [Hong, *et al.*, 1990]), training is rapid because learning is non-iterative [Specht, 1990], local minima are not present as is the case with neural-network training, and statistical optimality is assured for a sufficiently large training set. However, there are also some major disadvantages: (1) the required storage capacity grows linearly with training set size, and (2) the optical system

learns through interpolation between training vectors rather than through extraction of features relevant to the learning task. Disadvantage #2 is particularly important, because it means that this type of system never learns which combinations of input vector components are most important. This weakness is related to the "curse of dimensionality" [Bellman, 1961], and it implies that a very large number of training samples may have to be provided to the system in order to obtain satisfactory performance.

Adaptive neural-network learning algorithms offer a possible solution to these problems. Rather than directly storing the training set, adaptive neural algorithms adjust the network weights in order to approximate the desired input-output mapping. Typically the number of weights provided to the network does not scale linearly with training set size, as is the case for nonparametric statistical estimation methods. Thus the network is forced to use its weights parsimoniously, in the sense that it tries to learn the input-output mapping using fixed hardware by extracting those features from the training inputs that are most relevant to the learning task.

Photorefractive materials are currently the most promising holographic media for adaptive neural networks, in part because the dynamic nature of these volume materials allows for real-time interconnection updating. Unfortunately, the physics which governs photorefractive interconnection updates is not entirely consistent with the outer-product form common to many neural network learning algorithms; in a photorefractive material a decay effect causes previously written interconnections to be partially erased during each update [Psaltis *et al.*, 1988]. It was recently shown that this weight decay effect can prevent both the perceptron and backpropagation algorithms from converging to an acceptable solution

[Hsu *et al.*, 1993] [Petrisor *et al.*, 1995]. In this chapter, a technique we call *gain and exposure scheduling* is presented. This technique theoretically eliminates the effect of photorefractive weight decay from the outer-product (Hebbian) class of neural network learning algorithms, by iteratively increasing the spatial light modulator (SLM) transfer function gain and decreasing the weight update exposure time. Thus, the SLM's used for decay-compensated neural learning must have the flexibility of *variable input gain*, as was also the case for the PDF estimation architectures described in the previous chapter.

As will be shown in more detail below, an essential element in the analysis of photorefractive learning is the distinction between *physical representation* and *neural representation.* As a specific example, in previous articles on photorefractive neural networks [Psaltis *et al.*, 1988] [Owechko, 1993] the authors have treated synonymously the concepts of photorefractive diffraction efficiency (a physical implementation) and neural network interconnection weight (an abstract neural quantity). We will show that a clear distinction between these quantities is necessary for development of the gain and exposure scheduling technique. Thus, this chapter will begin with a physical description of photorefractive grating formation, together with its quantitative relationship with a neural network interconnection weight update. The scheduling procedure is first described for the conventional single coherent source (SCS) architecture, and the next section addresses gain and exposure scheduling as it applies to the more complicated Incoherent/Coherent (I/C) architecture [Jenkins *et al.*, 1990] [Asthana *et al.*, 1993]. Simulation results will then be presented which verify that the scheduling procedure results in improved learning performance.

4.2 SCS gain and exposure scheduling theory

The most common type of neural network weight update is known as a Hebbian [Rumelhart and McClelland, 1986] or outer-product update rule. This class of weight updates is defined by

$$w_{ik}(n+1) = w_{ik}(n) + \alpha\delta_i(n+1)x_k(n+1), \tag{4.1}$$

where w_{ik} is the connection weight from neuron k to neuron i, α is the (fixed) learning rate, n is the iteration number, and neural signals x_k and δ_i are algorithm-dependent. The class of algorithms to which Eq. (4.1) applies includes most of the widely used neural algorithms, such as backpropagation, perceptron, Hopfield, and LMS (Widrow-Hoff).

The update given by Eq. (4.1) can be optically implemented using spatial light modulators and a photorefractive crystal, as shown in Fig. 4.1. This figure shows the SCS physical interconnection geometry considered in this chapter for one layer of a feed-forward neural network. (For a multi-layer network, the input SLM can be the output SLM of the previous layer.) During photorefractive recording the shutter is open. Beams from both the training and input SLM's interfere in the photorefractive crystal to update previously written interconnection gratings. The grating update equation differs from Eq. (4.1) because of a decay term that models the partial erasure of previously recorded gratings [Owechko, 1993] [Anderson, 1987],

$$\tilde{\eta}_{ik}(n+1) = \tilde{\eta}_{ik}(n)(1-\beta_{n+1}) + [K\exp(j\phi)\beta_{n+1}]E_{\delta_i}(n+1)E^*_{x_k}(n+1). \tag{4.2}$$

In the above, E_{δ_i} and E_{x_k} are complex plane-wave amplitude representations of neural signals from the training and input planes, respectively. $\tilde{\eta}_{ik}$ represents complex-amplitude diffraction efficiency, which is defined as the ratio of the detected complex amplitude E_i^{det} (measured at the output SLM detection plane) to the input complex amplitude E_{x_k}. The phase term $\exp(j\phi)$ includes the effect of any constant phase shift caused by the photorefractive recording method and optical propagation. The real parameter K is dependent on total recording intensity, the photorefractive medium, and the optical recording geometry. Decay parameter β_n is related to the weight update exposure time Δt_n through the relation

$$\beta_n = 1 - \exp(-\Delta t_n/\tau), \tag{4.3}$$

where τ is the photorefractive grating formation (and grating erasure) time constant. We assume in the following that the total recording intensity is fixed throughout training, which implies that parameters K and τ are constants [Owechko, 1993]. In addition, if the training and input beams are all activated simultaneously there may be coherent recording cross-talk effects that are not considered in this analysis [Asthana *et al.*, 1993] (we will return to the subject of crosstalk in Sec. 4.3).

During photorefractive readout the shutter (Fig. 4.1) is closed, and each diffracted output plane-wave has an amplitude equal to a weighted coherent sum of the input amplitudes. It is assumed that the field strength is sufficiently weak so that any diffraction efficiency decay that occurs during readout is insignificant. The output SLM provides a (possibly) nonlinear transfer function to compute the neural activation function. It also supplies a *variable gain* that can be implemented either

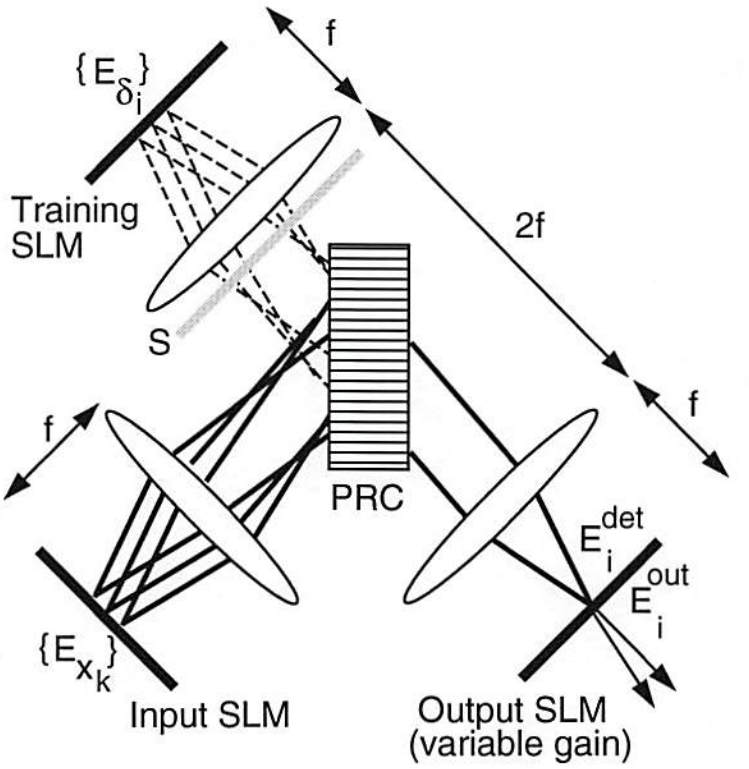


Figure 4.1: SCS interconnection geometry for a single neural network layer (one diffracted output beam shown); PRC, photorefractive crystal; S, shutter; f, focal length

by varying the SLM detection sensitivity or by varying the SLM readout intensity. In the following derivation, the output SLM uses coherent (field amplitude and phase) detection to distinguish between positive and negative inputs. In addition, the training and input SLM's use $[0, \pi]$ phase modulation to optically represent bipolar neural signals. Coherent detection and phase modulation are analytically convenient but difficult to implement. Therefore we will also show below that the results obtained can be adapted to the special case of a dual-rail encoded system that requires neither phase detection nor phase modulation.

The variable SLM gain, in conjunction with proper scheduling of the weight update exposure time, allows the ideal neural update rule given by Eq. (4.1) to be realized by the physical interconnection geometry shown in Fig. 4.1. To show this, we first relate electric field amplitude to neural signal level,

$$\delta_i(n) = E_{\delta_i}(n)/|E_r|, \qquad x_k(n) = E_{x_k}(n)/|E_r| \tag{4.4}$$

and amplitude diffraction efficiency to its corresponding neural interconnection weight,

$$w_{ik}(n+1) = G_{n+1}\tilde{\eta}_{ik}(n+1)\exp(-j\phi). \tag{4.5}$$

In the above, $|E_r|$ denotes the (constant) SLM-pixel readout field magnitude used during photorefractive recording, which corresponds to maximal neural activation assuming a maximum SLM reflectivity of one. The variables x_k, δ_i, w_{ik}, G_n, E_{x_k}, and E_{δ_i} are all *real* numbers. E_{x_k} and E_{δ_i} are constrained to be real by restricting SLM phase modulation to 0 and π rad (*i.e.*, any pair of point sources are either in-phase or π rad out of phase).

The parameter G_n denotes the field amplitude gain supplied by the output

SLM, as defined by its input-output transfer function, f_{SLM}:

$$E_i^{out}(n) = f_{\text{SLM}}(G_n E_i^{det}(n)) = f_{\text{SLM}}\left(G_n \sum_k E_{x_k}(n)\tilde{\eta}_{ik}(n)\right) \quad (4.6)$$

$$= f_{\text{SLM}}\left(|E_r'|\exp(j\phi)\sum_k x_k(n)w_{ik}(n)\right), \quad (4.7)$$

where E_i^{det} and E_i^{out} are the complex amplitudes that correspond to the SLM detected input and modulated output at neuron i during photorefractive readout, and $|E_r'|$ is the weak field-magnitude used during photorefractive readout. Fig. 4.2 diagrams the system-level SLM model corresponding to Eq. (4.6). This model is similar to Fig. 3.3 from the previous chapter, in which dual-rail intensities were used to modulate the output. Here we assume that the modulator has the capability for amplitude and phase detection (phase can be detected interferometrically), which eliminates the need for dual-rail detection. As with the SLM discussed in the previous chapter, the multiple opto-electronic transductions can be described at the system level by the optical-input/optical-output transfer function given by Eq. (4.6). Eq. (4.7) shows that transfer function $f_{\text{SLM}}(\cdot)$ is a scaled version of the *activation function* (squashing function) used at the neural-level description.

The mapping between the physical and neural network quantities given above can now be used to derive the constraints under which the ideal neural outer-product update rule is obtained. Substitution of Eqs. (4.2) and (4.4) into Eq. (4.5), while constraining the coefficients to match the desired form of Eq. (4.1), yields the relations

$$(G_{n+1}/G_n)(1-\beta_{n+1}) = 1 \quad (4.8)$$

$$\beta_{n+1}G_{n+1} = \alpha/(K|E_r|^2) = \beta_n G_n, \quad (4.9)$$

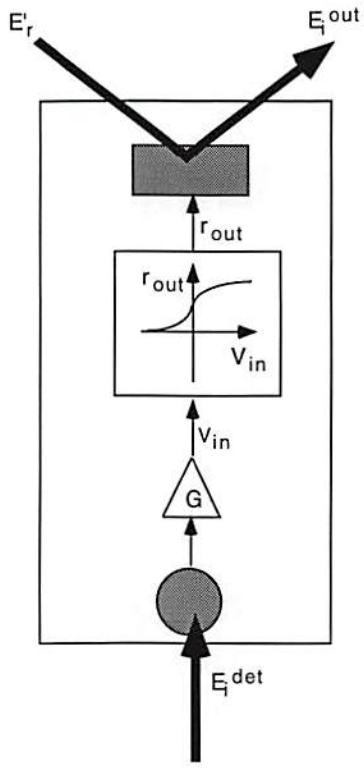


Figure 4.2: System-level SLM model

where the final equality in Eq. (4.9) is valid because $\alpha/(K|E_r|^2)$ is a constant. The combination of Eqs. (4.8) and (4.9) results in an iterative exposure schedule given by

$$\beta_{n+1} = \beta_n/(1+\beta_n), \tag{4.10}$$

which, together with Eq. (4.9), implies the closed form solutions

$$\beta_n = \beta_0/(1+n\beta_0) \tag{4.11}$$

$$G_n = G_0(1+n\beta_0) \tag{4.12}$$

$$G_0\beta_0 = \alpha/(K|E_r|^2). \tag{4.13}$$

Eqs. (4.11)–(4.13), together with Eq. (4.3), give the required gain and exposure schedule for decay-compensated weight updates for each neural network layer. The exposure schedule given by Eq. (4.11) is similar to those derived in previous articles, which considered the recording of holograms with equal diffraction efficiency [Blotekjaer, 1979] [Psaltis *et al.*, 1988]. However, when the required updates are generated with an iterative nonlinear neural algorithm, *both* gain and exposure scheduling must be used simultaneously.

The combination of Eqs. (4.13) and (4.12) yields

$$G_{n_f} = G_0 + (\alpha n_f)/(K|E_r|^2), \tag{4.14}$$

in which n_f is the number of iterations required for network convergence. Eq. (4.14) implies that a tradeoff exists between the required maximum SLM gain (G_{n_f}) versus the required dynamic range of the SLM gain (G_{n_f}/G_0), for the following

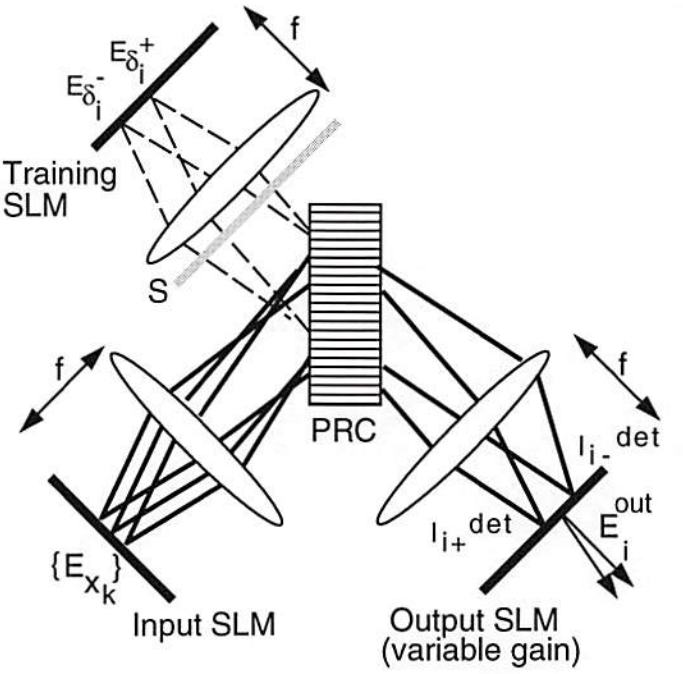


Figure 4.3: Dual-rail SCS interconnection geometry

reason. It can be shown that, in the "continuous-time limit" ($\alpha << 1$), the product (αn_f) depends only on the weight space path traversed by the network during learning; that is, the product is independent of the α, G_0, and β_0. Therefore, the final SLM gain G_{n_f} is minimized by setting the initial gain G_0 to be as small as possible. However, the SLM gain dynamic range requirement (G_{n_f}/G_0) is most severe (largest ratio) for small G_0. The simulation results presented below will verify the existence of this tradeoff.

4.2.1 Dual-rail representation

Because phase modulation and detection are difficult to accurately implement in practice, we also applied the gain and exposure schedule derived above to the special case of a dual-rail signal representation [Owechko, 1993], as shown in

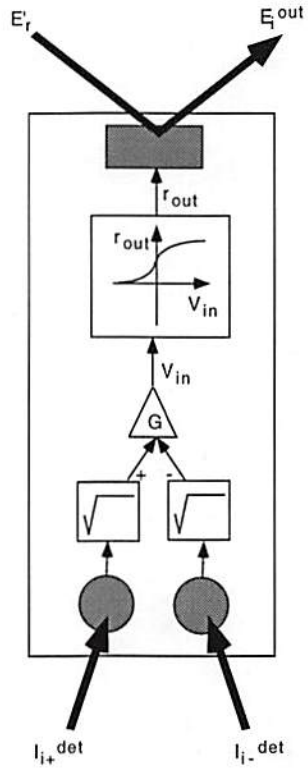


Figure 4.4: Dual-rail SCS model for SLM

Fig. 4.3 and Fig. 4.4. In such as system, each bipolar quantity is represented by two unipolar (positive) quantities, thus eliminating the need for phase modulation and detection. Each bipolar neural weight is represented using the *magnitudes* of two separate interconnection gratings according to the modified mapping

$$w_{ik}(n+1) = G_{n+1}(\left|\tilde{\eta}_{ik}^{+}(n+1)\right| - \left|\tilde{\eta}_{ik}^{-}(n+1)\right|), \tag{4.15}$$

where $\tilde{\eta}_{ik}^{+}$ and $\tilde{\eta}_{ik}^{-}$ are the amplitude diffraction efficiencies associated with the positive and negative components (respectively) of the bipolar connection weight. Correspondingly, at the output SLM each amplitude E_i^{det} is computed as the difference between the square roots of the dual-rail detected intensities I_{i+}^{det} and I_{i-}^{det}, according to

$$\begin{aligned} E_i^{det} &= \sqrt{I_{i+}^{det}} - \sqrt{I_{i-}^{det}} \\ &= \sqrt{\left|\sum_k E_{x_k}\tilde{\eta}_{ik}^{+}\right|^2} - \sqrt{\left|\sum_k E_{x_k}\tilde{\eta}_{ik}^{-}\right|^2} \\ &= \sum_k |E_{x_k}|\,(\left|\tilde{\eta}_{ik}^{+}\right| - \left|\tilde{\eta}_{ik}^{-}\right|), \end{aligned} \tag{4.16}$$

where Eq. (4.16) follows from the preceding expression because phase modulation is not used during photorefractive recording. The required subtraction and square root operations are assumed to be internally computed (electronically) by the SLM. The mapping between neural signals and their optical representation is modified according to

$$\delta_i(n) = (\left|E_{\delta_i}^{+}(n)\right| - \left|E_{\delta_i}^{-}(n)\right|)/|E_r|, \qquad x_k(n) = |E_{x_k}(n)|\,/|E_r| \tag{4.17}$$

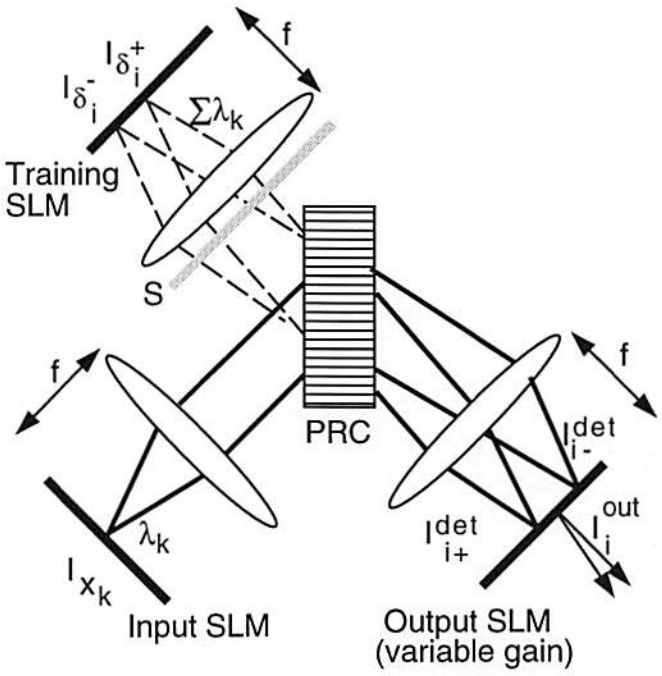


Figure 4.5: I/C interconnection geometry

where $\left|E_{\delta_i}^+\right|$ updates $\left|\tilde{\eta}_{ik}^+\right|$ and $\left|E_{\delta_i}^-\right|$ updates $\left|\tilde{\eta}_{ik}^-\right|$. The input SLM does not require a dual-rail representation because all external inputs to the network are restricted to be nonnegative, as are all neuron outputs. (That is, the *input* to each neuron is bipolar, but the output is unipolar). Following a procedure similar to that used for the previous derivation (taking the magnitude of Eq. (4.2) with appropriate superscripts, substituting into Eq. (4.15) and comparing to Eq. (4.1) combined with Eq.(4.17)), it can be shown that the same gain and exposure schedule is obtained as that given by Eqs. (4.8)–(4.13).

4.3 I/C gain and exposure scheduling theory

The Incoherent/Coherent (I/C) architecture [Jenkins *et al.*, 1990] [Asthana *et al.*, 1993] was recently proposed as an alternative to the fully coherent

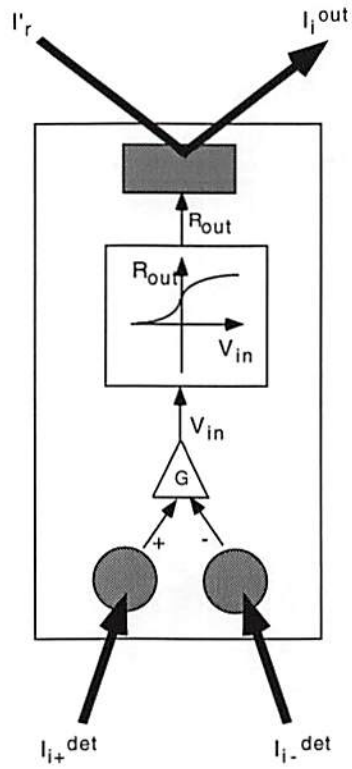


Figure 4.6: I/C model for SLM

(SCS) architecture [Psaltis *et al.*, 1988]. Fig. 4.6 shows the interconnection geometry for one variant of the I/C architecture. At the input SLM, each point source is individually coherent but mutually incoherent with respect to the other input SLM point sources. This mutual incoherence prevents undesirable interconnection gratings from being written in the crystal. At the training SLM, each point source represents a superposition of all sources used at the input SLM; therefore, a grating can be written between any training SLM pixel and any input SLM pixel. Because the input sources are mutually incoherent, during holographic readout each diffracted output beam has an intensity equal to a weighted summation (weight proportional to *intensity* diffraction efficiency) of the input intensities.

Unfortunately, one price to be paid for the crosstalk reduction in the I/C architecture is that the standard Hebbian rule given by Eq. (4.1) cannot be implemented in a photorefractive crystal. A revised version of the Hebbian rule can be implemented, however, as given by

$$\begin{aligned} w_{ik}(n+1) &= w_{ik}(n) + (\alpha^2/4)\delta_i(n+1)x_k(n+1) \\ &+ \alpha\left(\sqrt{\delta_i^+(n+1)w_{ik}^+(n)} - \sqrt{\delta_i^-(n+1)w_{ik}^-(n)}\right)\sqrt{x_k(n+1)}, \end{aligned} \quad (4.18)$$

in which δ_i^+ is the positive component of δ_i [$\delta_i^+ = (1/2)(\delta_i + |\delta_i|)$] and δ_i^- is the negative component of δ_i [$\delta_i^- = (1/2)(\delta_i - |\delta_i|)$]. The two unipolar weights w_{ik}^+ and w_{ik}^- are related to the bipolar weight w_{ik} through the relation $w_{ik} = w_{ik}^+ - w_{ik}^-$. Although this is clearly not equivalent to the ideal Hebbian update rule of Eq. (4.1), it can be shown that gradient-descent neural network algorithms based on the true Hebbian update will still converge with the revised update. (See Appendix 7 for more details).

The form of the revised Hebbian rule given by Eq. (4.18) is caused by the new mapping between neural network quantities and their optical representation in the I/C architecture. For this architecture, neural signals are optically represented using the relations

$$\delta_i^+(n) = I_{\delta_i}^+(n)/I_r \qquad \delta_i^-(n) = I_{\delta_i}^-(n)/I_r \qquad x_k(n) = I_{x_k}/I_r, \tag{4.19}$$

in which I_r is the SLM pixel readout intensity. The neural interconnection weights are related to their corresponding dual-rail diffraction efficiencies through the relation

$$w_{ik}(n+1) = G_{n+1}^I \left(|\tilde{\eta}_{ik}^+(n+1)|^2 - |\tilde{\eta}_{ik}^-(n+1)|^2 \right). \tag{4.20}$$

The relationship given above is similar to that used in the dual-rail SCS representation [Eq. (4.15)]. However, in this case it is the *intensity* diffraction efficiency $|\tilde{\eta}^2|$, rather than the *amplitude* diffraction efficiency $\tilde{\eta}$, that determines the interconnection weight. The SLM intensity gain G^I is defined by its input-output transfer function, $f_{\mathrm{SLM}}(\cdot)$:

$$\begin{aligned} I_i^{out} &= f_{\mathrm{SLM}}[G^I(I_{i+}^{det} - I_{i-}^{det})] \\ &= f_{\mathrm{SLM}}[G^I \sum_k I_{x_k}(|\eta_{ik}^+|^2 - |\eta_{ik}^-|^2)] \\ &= f_{\mathrm{SLM}}(I_r \sum_k x_k w_{ik}). \end{aligned} \tag{4.21}$$

The mapping between the physical and neural network quantities given above can now be used to derive the constraints under which the revised Hebbian update rule given by Eq. (4.18) is obtained. Substitution of Eq. (4.2) (taking the magnitude squared and adding appropriate superscripts) and Eq. (4.19) into Eq. (4.20),

while constraining the coefficients to match the form of Eq. (4.18), yields the following relations:

$$(G^I_{n+1}/G^I_n)(1-\beta_{n+1})^2 = 1 \tag{4.22}$$

$$G^I_{n+1}\beta^2_{n+1}(KI_r)^2 = \alpha^2/4 = G^I_n\beta^2_n(KI_r)^2 \tag{4.23}$$

$$2(KI_r)\beta_{n+1}(1-\beta_{n+1})\left(G^I_{n+1}/\sqrt{G^I_n}\right) = \alpha. \tag{4.24}$$

The combination of Eqs. (4.22) and (4.23) results in the same iterative exposure schedule derived earlier,

$$\beta_{n+1} = \beta_n/(1+\beta_n) \tag{4.25}$$

which together with Eq. (4.23) implies the closed form solutions

$$\beta_n = \beta_0/(1+n\beta_0) \tag{4.26}$$

$$G^I_n = G_0(1+n\beta_0)^2 \tag{4.27}$$

$$G^I_0\beta^2_0 = [\alpha/(2KI_r)]^2. \tag{4.28}$$

It can be shown that the third constraint given by Eq. (4.24) is automatically satisfied when the first two constraints are satisfied.

The above calculations reveal a surprising result: even though the physics governing the I/C architecture differs significantly from that governing the SCS architecture, and the form of the Hebbian updates are substantially different [Eq. (4.1) versus Eq. (4.18)], the gain and exposure schedules turn out to be basically the same in both cases. (The intensity gain G^I_n rises quadratically with n in the I/C architecture, while G_n rises linearly with n in the SCS architecture. This means

that the gain schedules are actually identical, because the former represents *amplitude* gain and the latter represents *power* gain.)

4.4 Simulation results

We simulated the derived gain and exposure schedules for both the dual-rail SCS and I/C architectures. In both cases, we used backpropagation as the learning algorithm and XOR (2 input units, 4 hidden units, 1 output unit) as the test problem [Rumelhart and McClelland, 1986].

4.4.1 SCS dual-rail simulations

For the dual-rail SCS simulations, we used a sigmoidal SLM input-output transfer function given by

$$E_i^{out} = f_{\mathrm{SLM}}(G_n E_i^{det}) = |E_r'|/[1 + \exp(-4G_n E_i^{det}/|E_r'|)], \tag{4.29}$$

where $|E_r'|$ is the weak field magnitude used during photorefractive readout. The detected field E_i^{det} was computed using the dual-rail representation indicated in Eq. (4.16). The material-dependent parameter $(K|E_r|^2)$ in Eq. (4.13), which is inversely proportional to the number of neurons per layer, was set equal to (1/4).

Fig. 4.7 shows typical dual-rail simulation results for network error as a function of iteration number, both with and without the scheduling procedure. When scheduling was not used (exposure time and SLM gain were held constant), the network did converge to a steady-state in weight space, but the final network did not solve the XOR problem. The simulation which used gain and exposure

scheduling converged to an acceptable solution, and it was identical to decay-free (no decay term in weight update equation) backpropagation training at each iteration.

Table 4.1 shows statistically averaged simulation results that demonstrate the advantage of gain and exposure scheduling. As shown in the table, the backpropagation simulations that did not use gain and exposure scheduling converged only when the gain was sufficiently high. When scheduling was used, the networks converged at a rate of 100% regardless of initial gain. The number of iterations required for network convergence was also reduced in the scheduled simulations, because the effect of weight decay (which alters the optimal gradient-descent weight updates) was eliminated. Furthermore, the average final gain for the scheduled simulations was significantly smaller (when the initial gain was set equal to one) than the gain required for 100% convergence in the constant gain simulations. Thus, after training is complete and the network is used in an optical system, a network trained with scheduling would require a smaller gain (on average) than one trained without scheduling. The data also verify the tradeoff discussed at the end of Sec. (4.2); the final gain is minimized when $G_0 = 1$, but the ratio G_{n_f}/G_0 is largest for this value of G_0.

4.4.2 I/C simulations

The gain and exposure scheduling simulations for the I/C architecture were quite similar to those applied to the dual-rail SCS architecture. For the I/C case, the SLM transfer function is an input intensity – output intensity sigmoid, given by

$$I_i^{out} = f_{\mathrm{SLM}}[G_n^I(I_{i+}^{det} - I_{i-}^{det})]$$

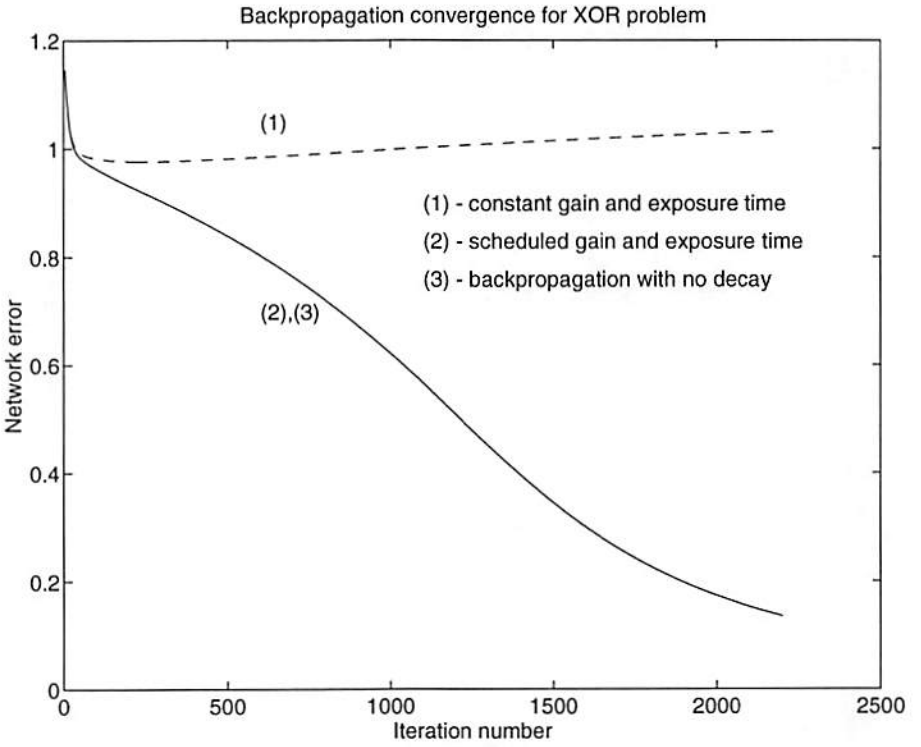


Figure 4.7: Backpropagation for XOR (SCS, $\alpha = 0.025$, $G_0 = 100$). (1) constant gain and exposure; (2) scheduled gain and exposure; (3) weight update with no decay term.

Table 4.1: **Backpropagation XOR convergence[a] (SCS architecture)**

		No Scheduling		Scheduling		
α	G_0	$C(\%)$	$<n_f>$	$C(\%)$	$<n_f>$	$<G_{n_f}>$
0.1	1	0	–	100	636	257
0.1	350	45	1966	100	637	605
0.1	500	95	1286	100	675	770
0.1	1000	100	780	100	648	1259
0.2	1	0	–	100	426	342
0.2	350	39	1394	100	426	691
0.2	500	93	781	100	394	815
0.2	1000	100	446	100	424	1339

[a] A network converged if the absolute difference between the target and actual output was less than 0.2 for all input patterns, allowing a maximum of 4000 training iterations. Results were averaged over 100 trials with random initial weights. Gain (and decay) parameters were held constant at initial values when scheduling was not used. α, learning rate; G_0, initial gain; C, convergence rate; $<n_f>$, avg. number of iterations; $<G_{n_f}>$, avg. final gain.

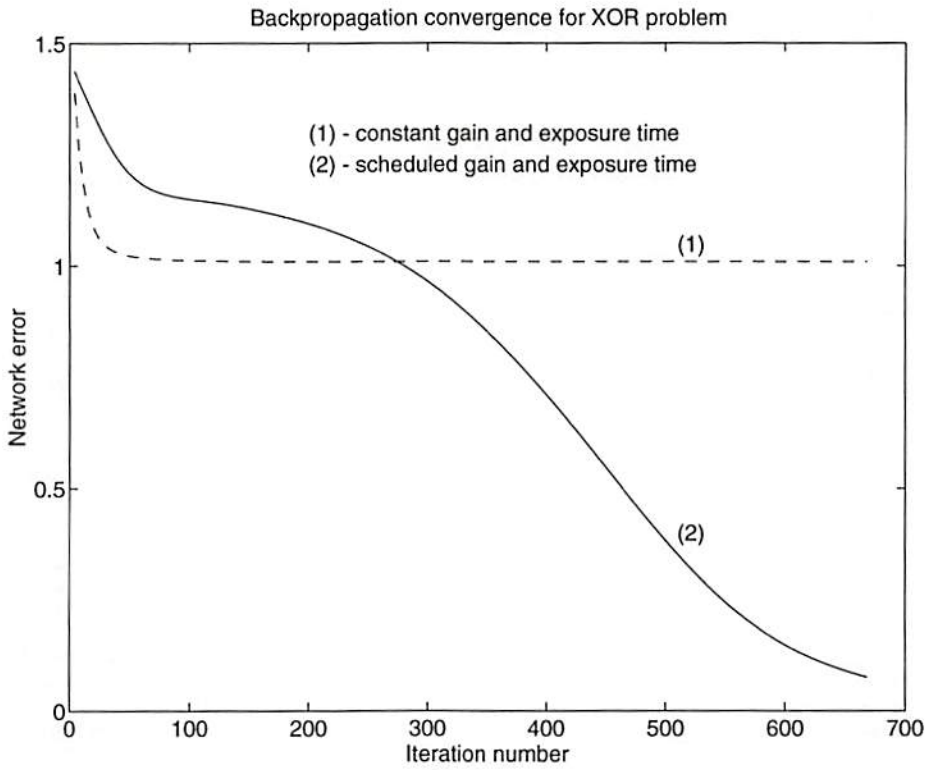


Figure 4.8: Backpropagation for XOR (I/C, $\alpha = 0.025$, $G_0^I = 1$). (1) constant gain and exposure; (2) scheduled gain and exposure

$$I_i^{out} = I_r'/[1 + \exp\left[-4G_n^I(I_{i+}^{det} - I_{i-}^{det})/I_r'\right], \tag{4.30}$$

in which I_r' is the weak intensity used during photorefractive readout.

Fig. 4.8 shows typical simulation results for network error as a function of iteration number, both with and without the scheduling procedure. The results are quite similar to the SCS case discussed above. When scheduling was not used, the network often settled into an unacceptable local minimum; when gain and exposure scheduling was used, unacceptable local minima were extremely rare.

Table 4.1 shows statistically averaged simulation results that demonstrate the advantage of I/C gain and exposure scheduling. As was the case for the SCS case, the backpropagation simulations that did not use gain and exposure scheduling

Table 4.2: **Backpropagation XOR convergence[a] (I/C architecture)**

		No Scheduling		Scheduling		
α	G_0	$C(\%)$	$<n_f>$	$C(\%)$	$<n_f>$	$<G_{n_f}>$
0.025	1	0	–	96	794	2038
0.025	2500	0	–	96	795	8421
0.025	4900	68	1861	94	915	12616
0.025	10000	100	1228	95	814	20150
0.05	1	0	–	97	473	3455
0.05	2500	0	–	97	472	10563
0.05	4900	64	1383	97	496	15544
0.05	10000	100	470	96	470	22591

[a]A network converged if the absolute difference between the target and actual output was less than 0.2 for all input patterns, allowing a maximum of 4000 training iterations. Results were averaged over 100 trials with random initial weights. Gain (and decay) parameters were held constant at initial values when scheduling was not used. α, learning rate; G_0, initial gain; C, convergence rate; $<n_f>$, avg. number of iterations; $<G_{n_f}>$, avg. final gain.

converged only when the gain was sufficiently high; when scheduling was used, the networks converged at a rate of nearly 100%. However, gain and exposure scheduling for the SCS architecture resulted in *exactly* 100% convergence. Apparently the weight update given by Eq. (4.18) is slightly more likely to lead the network into a local minimum of the error surface. One surprising simulation result is that, if the SLM gain is sufficiently high, the weight updates *with* decay can converge at a rate of 100%. This indicates that a small amount of weight decay is actually beneficial for the I/C architecture. (A similar gain and exposure schedule can be derived that, rather than completely eliminating the effect of weight decay, allows the decay rate to be set at a user-specified value.)

4.5 Discussion

The scheduling algorithm is applicable to any neural network learning rule that can be mapped into the form of Eq. (4.1) (*e.g.* backpropagation, Widrow-Hoff, perceptron, Hopfield). If the network architecture uses step-function neural activation functions, as is the case for perceptron and some Hopfield networks, only the exposure schedule of Eq. (4.11) is needed to eliminate the effect of photorefractive diffraction efficiency decay. Gain scheduling is unnecessary because in this special case $f_{\mathrm{SLM}}(\cdot)$ is a step function (*i.e.* the SLM output depends only on the *sign* of the detected input), which is invariant with respect to any scaling of its input argument. It has recently been shown that, without the use of an exposure schedule, the perceptron training algorithm will not converge unless the (constant) exposure energy is sufficiently small [Hsu *et al.*, 1993]. Furthermore, even when it does converge, the perceptron with decay requires more training iterations than a

decay-free perceptron. It is therefore advantageous to use the exposure schedule of Eq. (4.11), as this guarantees convergence in the same number of iterations as a decay-free perceptron (for the SCS architecture).

Although the gain and exposure schedule theoretically eliminates the effect of weight decay regardless of the number of iterations required for network convergence, there is a practical limit determined by the minimum tolerable diffraction efficiency and/or the maximum allowable SLM gain. Gain $G_n \rightarrow \infty$ as $n \rightarrow \infty$ (Eq. (4.12)), which implies that $|\tilde{\eta}_{ik}(n)| \rightarrow 0$ because the neural weights are finite (Eq. (4.5)). One way to alleviate both the diffraction efficiency and gain problems is to periodically refresh the photorefractive gratings during the gain and exposure scheduling procedure [Qiao *et al.*, 1991] [Brady *et al.*, 1990]. After each refresh operation the SLM gain would be reduced by a factor equal to the refresh gain, thereby keeping all neural weights unchanged while increasing diffraction efficiency. Thus, the combination of gain and exposure scheduling together with periodic diffraction efficiency refreshing could provide decay-compensated neural network performance for an arbitrarily large number of training iterations.

Chapter 5

A 3-D photonic multichip-module neural network: hardware architecture

5.1 Introduction

In the previous chapters of this dissertation, several examples of scalable photonic neural network architectures that use volume holographic weight storage were presented. In the case of the Continuous-k Nearest Neighbor neural network, the emphasis was on the new probability density estimation algorithm together with a mapping between the new algorithm and a volume holographic implementation. The gain and exposure scheduling technique was also designed for use with a volume holographic medium, where in this case the dynamics of photorefractive grating formation were exploited in order to implement the outer-product update common to many neural-network learning algorithms.

This and the following chapter discuss a different type of scalable photonic neural network. Whereas the emphasis in the previous chapters was primarily on the use of a 3-D optical interconnection medium for neural-network processing, in these two chapters the emphasis is on 3-D packaging of multiple electronic

and photonic components into a compact neural-network structure. We call such a structure a "3-D Photonic Multichip-Module Neural Network" (3-D PMCM Neural Network). Recent technological advances in the hybridization of GaAs optical modulators with high-density silicon CMOS circuitry [Goossen *et al.*, 1995] [Kyriakakis *et al.*, 1995] [Worchesky *et al.*, 1996] have enabled the design of dense high-bandwidth optoelectronic devices suitable for such 3-D neural-network processors. Progress has also been rapid in the area of Diffractive Optical Element (DOE) [Jahns, 1994] and microlens [Oikawa and Hamanaka, 1994] fabrication. The availability of DOE and microlens foundries enable the fabrication of compact optical interconnection systems that can be used for neural-network computations. These DOE-based interconnection systems are designed on a digital computer, and they are fabricated using the same photolithographic techniques that are standard in the microelectronic semiconductor industry. Thus, the 3-D PMCM neural network architecture leverages off, and can benefit from improvements in, state-of-the-art manufacturing processes.

In the semiconductor industry, the term "Multichip Module" (MCM) refers to a package consisting of multiple bare die (chips) that are attached directly to a common silicon or ceramic substrate [Licari, 1995]. This style of packaging avoids the use of a chip carrier and its associated high-capacitance I/O pins. The result is that significantly lower capacitance, higher bandwidth, and more finely spaced electrical interconnects can be fabricated in comparison with a conventional printed circuit board. Typically, the chips in an MCM are packaged in a 2-D (planar) layout, although there are development efforts underway in the implementation of 3-D electronic MCM's [Garvin, 1995].

Several research groups are investigating the use of free-space optical interconnections to replace or augment the metallic interconnections used in conventional planar multichip modules. Optical interconnections offer the potential of higher bandwidth, lower power consumption, and lower crosstalk than their electronic counterparts when the interconnect length exceeds a minimum break-even point [Feldman *et al.*, 1988]. At Bell Labs, Jurgen Jahns *et al.* have worked for several years on "planar optics" imaging systems that can be used for multichip module communication (*e.g.*, see [Jahns, 1994]). At the University of California at San Diego (UCSD), Fan *et al.* have developed algorithms for the placement of processing elements in an optically-interconnected planar multichip module [Fan *et al.*, 1995]. At the University of North Carolina, Feldman *et al.* have developed a prototype module that incorporates GaAs laser array chips and silicon CMOS chips which are flip-chip bonded onto a transparent MCM substrate; comparisons between optical interconnects and electrical interconnects show that for a 5-cm link the dissipated power can be reduced by at least an order of magnitude (for data rates between 700 MHz and 4 GHz) when optical interconnects are used [Feldman *et al.*, 1994].

The above research addresses point-to-point optical interconnections between chips in a planar multichip module. In this dissertation, we analyze a compact optical interconnection system for a stack of chips in a 3-D multichip module. Related research into 3-D photonic structures from other groups is briefly summarized in the following. Researchers at Georgia Tech are investigating optically-interconnected stacks of chips; they are studying digital point-to-point

interconnection architectures for a 3-D mesh-connected parallel image processor [Wills *et al.*, 1995]. Veldkamp *et al.* have discussed 3-D photonic structures in the context of multi-layered focal planes that are inspired by biological visual processing systems [Veldkamp, 1993]. At UCSD, Yayla *et al.* have recently demonstrated a prototype 3-D optically interconnected neural network [Yayla *et al.*, 1994]. Their system differs from that discussed in this dissertation in that it utilizes neither DOE nor optoelectronic hybridization technology.

The remainder of this chapter discusses the design and initial tests of our 3-D PMCM neural-network architecture. In Sec. 5.2, an overview is given in which each layer of the 3-D PMCM neural network is briefly described. Sections 5.3 and 5.4 discuss the functionality of the 3-D PMCM input-layer chip and the associated mapping between neural-network signals and their optical representations. Section 5.5 discusses details of the design, simulation, and testing of the input-layer CMOS VLSI chip. In Section 5.6, the design and fabrication of the 3-D PMCM neural-network diffractive optical interconnection system is described in some detail. Finally, Sec. 5.7 concludes with a discussion and possible extensions of this work.

5.2 Overview of 3-D PMCM neural-network hardware

A 3-D PMCM neural network consists of several stacked optical and optoelectronic layers, as shown in Fig. 5.1 (a). The figure shows a vertical cross section of a three-dimensional stacked structure. In Fig. 5.1 (b), this same cross

section is represented as a standard 2-layer feedforward neural network architecture, consisting of a layer of input signals, a hidden layer with sigmoidal activation functions, and a single linear output unit. An overview of the complete multi-layer stack is given below, but this chapter focuses primarily on the design and fabrication of the bottom-most chip ("photonic signal-shifting chip") and the diffractive optical layers ("microDOE" and "microlens" arrays). The "optical power bus" and "programmable neural-network chip" have not yet been designed for use in a 3-D PMCM neural network, although preliminary versions of these devices have been previously demonstrated [Demars, 1997] [Asthana *et al.*, 1990].

The Si/GaAs hybrid chip at the bottom of the stack serves as the input layer of the network. A 2-D plane of electronic signals representing the input-pattern information, such as image pixels or Fourier descriptors of audio waveforms, enter the PMCM at the bottom of the stack and are converted into optical signals using reflective GaAs modulators that have been flip-chip bonded onto a silicon electronic chip [Kyriakakis *et al.*, 1995] [Ananthanarayanan *et al.*, 1995] [Goossen *et al.*, 1995]. This input layer is a key component of the PMCM for two reasons. First, it converts the electrical input signals into optical signals that can be processed by the diffractive optical elements. Second, the input layer circuitry can provide the capability for the shifting of a temporal or spatial input signal during neural network processing. As will be described in Sec. 5.3, input signal shifting allows the photonic neural network to analyze a scrolling window of the input signal, thereby allowing a relatively small neural network to analyze a much larger 1-D or 2-D input signal by temporally multiplexing the neural network computation. To emphasize the importance of the input signal shifting functionality, we refer to this input-layer chip as the "photonic signal-shifting chip". In

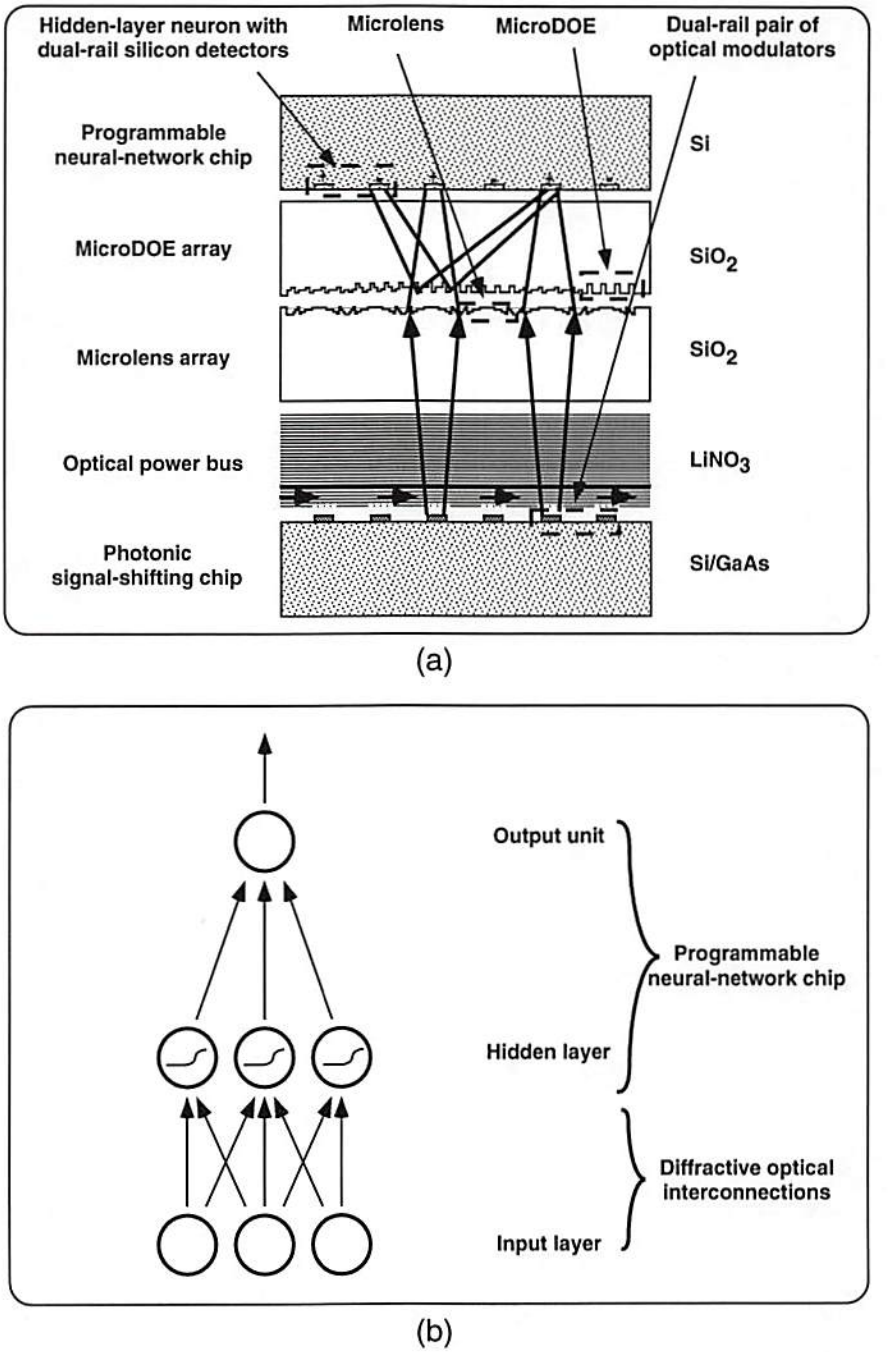


Figure 5.1: A 3-D PMCM neural network (a) hardware description and (b) functional architecture. In part (a), a 3-neuron cross-section of the PMCM stack is shown; a corresponding section of the functional architecture is shown in (b). For clarity, the readout beams for only 2 modulators are shown in (a). Substrate thicknesses are not drawn to scale.

Fig. 5.1, the photonic signal-shifting chip is operating in a "dual-rail" mode, where dual-rail refers to the use of two modulators to represent one input unit (which explains why there are 6 modulators in part (a) of the figure but only 3 neural input units in part (b)).

The optical modulators on the photonic signal shifting chip are read out with an external coherent source (*e.g.*, a laser diode) and an "optical power bus" [Ananthanarayanan *et al.*, 1995] that distributes the readout beam over the 2-D array of optical modulators. The optical power bus is a multiple-ribbed waveguide structure [Nishihara, 1989], in which each rib supplies the readout power to one row of optical modulators. A readout beam enters a rib of the optical power bus from the side of the PMCM stack, and light is outcoupled vertically to each of the optical modulators by blazed outcoupling gratings in the waveguiding layer [Nishihara, 1989]. At present, preliminary versions of the optical power bus (with unblazed gratings) have been fabricated in $LiNO_3$ and GaAs [Demars, 1997], but there are still technical issues (beyond the scope of this dissertation) to be resolved in the construction of a power bus suitable for use in a compact PMCM.

Each of the modulated beams addresses a microlens and a "micro-Diffractive Optical Element" (microDOE) [Huang, *et al.*, 1997] [Kuznia *et al.*, 1995]. A microDOE is a small-sized (*e.g.*, 125 μm on a side) diffractive optical element [Jahns, 1994] that splits a single coherent beam into an array (*e.g.*, 3x3) of fanout beams with (possibly) varying intensities. The phase-relief pattern etched into a microDOE determines the relative intensities of its fanout beams. These beams are focused onto a plane of detectors by a corresponding diffractive (as shown) or refractive microlens [Jahns, 1994] [Oikawa and Hamanaka, 1994], and the overlap

of the spots generated by nearby microlenses causes an incoherent superposition to occur at the detector plane (angularly-multiplexed beams add in intensity [Jenkins and Tanguay, 1992]). Thus, the microlens and microDOE arrays can optically implement one layer of local weighted neural interconnections, in which each interconnection weight is proportional to a corresponding fanout-beam diffraction efficiency. Because an *incoherent* superposition takes place at the detectors, only unipolar (positive) interconnection weights can be directly implemented using the microDOE array. However, the use of dual-rail pairs of detectors with an electronic subtraction capability (as described previously in Chapters 3 and 4) allows the interconnection weights to be effectively bipolar.

Finally, the programmable neural-network chip at the top of the PMCM stack implements the sigmoidal activation functions and the second layer of neural interconnections shown in Fig. 5.1 (b). This second layer can be fabricated using standard CMOS silicon VLSI processing; the dual-rail detectors and sigmoidal activation functions can be implemented with PN photodiodes and differential amplifiers (respectively) [Asthana *et al.*, 1990], while the programmable interconnection weights can be implemented using digital weight storage and (if necessary) multiplying-digital-to-analog converters (MDAC's) [Chiang and Chuang, 1991] [Yayla *et al.*, 1994]. By utilizing a hybrid approach in which a programmable electronic neural network follows a fixed diffractive optical layer, the strengths of both optical and electronic technologies are emphasized. The optical elements provide the high-bandwidth computation required for the first layer of interconnection weights, while the programmable electronic neural network provides the flexibility for the solution of a variety of pattern recognition tasks.

Although this and the next chapter focus on two-layer neural networks (one

layer of optical weights and one layer of electronic weights), it should be noted that the 3-D PMCM neural network architecture can be extended to more than two neural-network layers. For instance, if the programmable neural-network chip contained both detectors and (hybridized) modulators, and the chip were flipped right-side up, then another optical-power-bus/microDOE/microlens combination could be placed on top of it. In order to make this work, however, the silicon substrate needs to be transparent so that the incident light reaches detectors on the top surface of the chip. Unfortunately, silicon is not transparent to light at the operating wavelength of the GaAs/AlGaAs modulators on our photonic signal-shifting chip (850 nm), but GaAs/InGaAs modulators that operate at a more suitable wavelength are also available [Ananthanarayanan *et al.*, 1995].

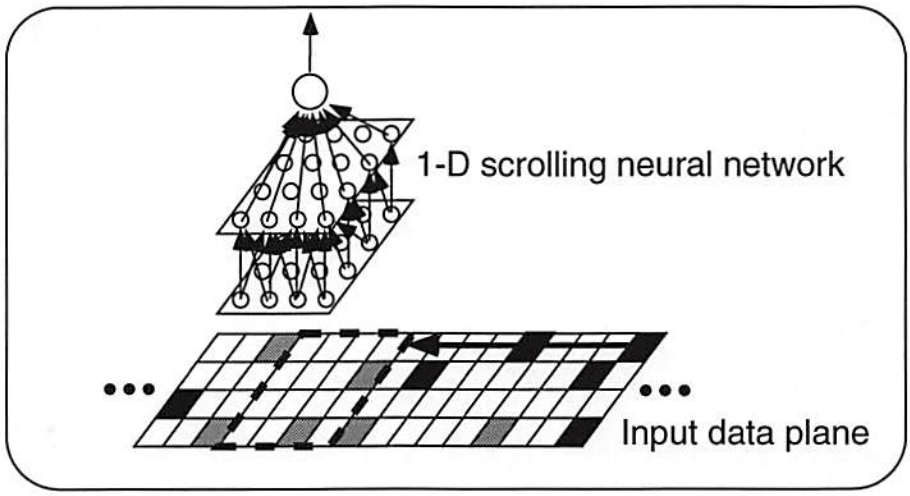
5.3 Photonic signal-shifting chip functionality

A key element of the 3-D PMCM neural-network architecture is the photonic signal-shifting chip. As discussed above, this chip combines two functions: it converts electronic input signals into optical signals for photonic neural network processing, and it also provides the signal-shifting capability that allows a relatively small neural network to analyze a much larger input plane through temporal multiplexing. Fig. 5.2 shows how this temporal multiplexing proceeds. In Fig. 5.2 (a), a neural network scrolls from right to left over the input data plane. At each clock cycle the neural network moves one column to the left, analyzes the current input window through feed-forward neural network processing, and the output unit yields a final result. We refer to this type of temporal multiplexing as "1-D scrolling", in that the input window moves in only one direction over the input data plane (although the input window is itself 2-D). In 2-D scrolling

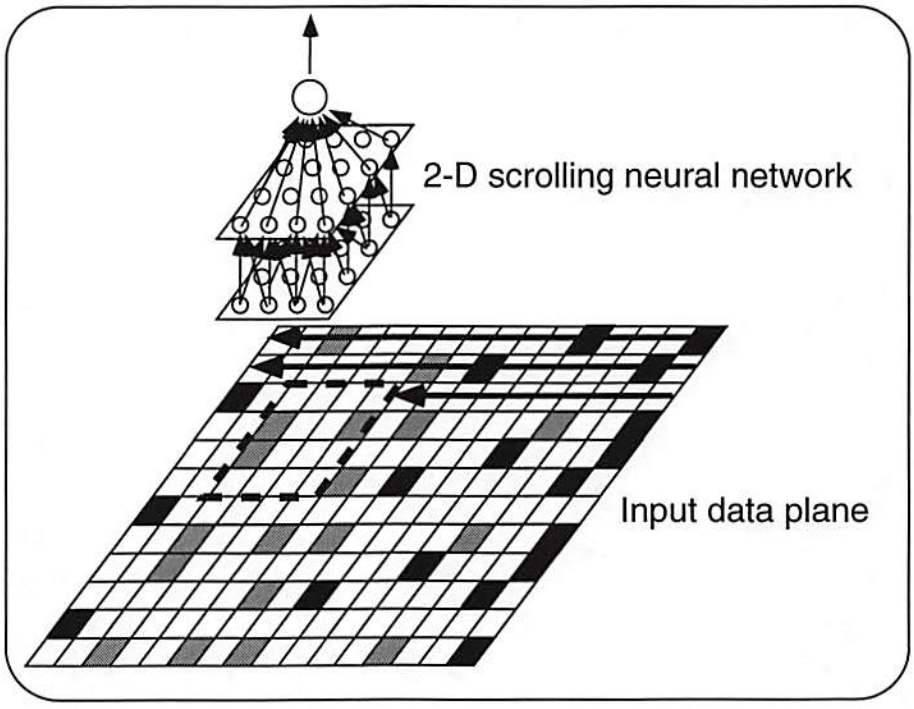
applications (Fig. 5.2 (b)), the neural-network input window scrolls, in both dimensions, over every possible location of the 2-D input data plane.

Typically, applications that use 1-D neural network scrolling require analysis of a parallel set of 1-D signals, while those that use 2-D scrolling involve image analysis. A good example of a neural-network architecture that can take advantage of 1-D scrolling is the Time Delay Neural Network (TDNN), which has been used for speech recognition algorithms that have yielded impressive results [Waibel *et al.*, 1989]. For a TDNN, the horizontal axis of the input data plane represents time, and the other axis (into the page) typically represents various time-windowed Fourier components of a speech signal. Image processing applications, such as automatic target recognition (ATR) or object tracking systems, can make use of 2-D neural network scrolling. For example, the eye detection system discussed in the next chapter requires a neural network to analyze every possible window of an input image. At each clock cycle the neural network reports whether or not the current input window contains a centered human eye.

The photonic signal-shifting chip described in this chapter can directly implement 1-D neural network scrolling, as shown in Fig. 5.3. In the figure, the three main layers of a 3-D PMCM neural network are depicted: a photonic signal-shifting chip, a microDOE array, and a programmable neural-network chip. As shown, n electronic signals $\{D_i[t] : i = 1 \ldots n\}$ enter the left side of the photonic signal-shifting chip at discrete time-step t. After a delay of m clock cycles, where m is the number of columns in the photonic signal-shifting chip, the same n signals electronically exit the chip. The entire 2-D array of optical modulators on top of the photonic signal-shifting chip is read out at every clock cycle, thereby allowing the vertically stacked opto-electronic neural network to analyze an $n \times m$



(a)



(b)

Figure 5.2: Scrolling neural network: (a) 1-D scrolling, (b) 2-D scrolling

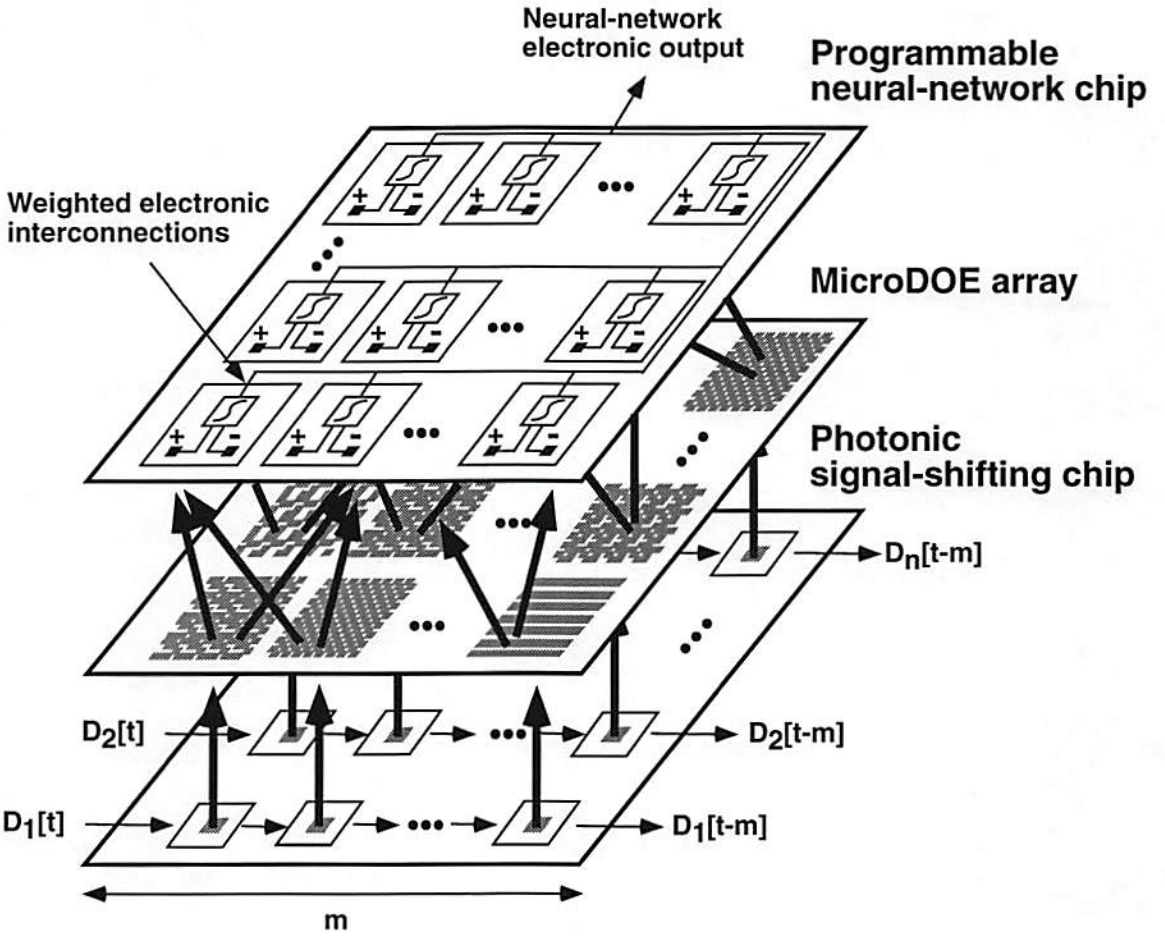


Figure 5.3: Photonic signal-shifting chip provides a scrolling input to the 3-D PMCM neural network. The n electronic input signals are delayed by m clock cycles before they exit the chip. The entire 2-D array of optical modulators is read out each clock cycle.

window of the input data that scrolls in one dimension.

A comparison between the electronic input/output (I/O) bandwidth and the optical I/O bandwidth in the system shown in Fig. 5.3 demonstrates the usefulness of photonics in applications that require analysis of a scrolling 2-D plane. At each clock cycle, n pixels enter the chip and n pixels exit the chip, yielding a data rate of $2nf$ pixels/sec of electronic I/O for a clock frequency of f Hz. The data rate that needs to be transmitted to the neural network analyzing the input

data is much larger. Since the entire 2-D array of modulators is read out at each clock cycle, mnf pixels/sec are transmitted to the scrolling neural network. The photonic signal-shifting chip we fabricated has $m = 20$ columns, which implies that the optical I/O bandwidth is an order of magnitude greater than the electronic I/O bandwidth. Thus, the signal-shifiting application is very well suited to photonic implementation; the relatively modest electronic bandwidth is handled by conventional I/O pads that are restricted to the periphery of the chip, while the much more I/O intensive task of transmitting 2-D planes of data is handled by the 2-D array of optical modulators.

Either analog or digital electronic circuitry can be used to implement the signal-shifting functionality. The chip that we fabricated is digital: the input signals $\{D_i[t] : i = 1 \ldots n\}$ are binary valued, the signal-shifting circuitry is composed of digital D flip-flops, and the modulators are driven to either the minimum-reflectivity (off) or maximum-reflectivity (on) state. In our case, the restriction to binary-only input signals is undesirable, but we were limited by the capabilities of the SEED modulators and the digital CMOS VLSI processing to which we had access. In the future, it would be very useful to design an analog version of the photonic signal-shifting chip. Such a chip could use analog CCD technology to provide the signal-shifting functionality, together with analog circuitry to drive the optical modulators.

In order to implement the 2-D scrolling necessary for image processing applications, additional electronic hardware that interfaces with the photonic signal-shifting chip is needed. This additional electronic hardware will be called the "2-D scrolling module" (Fig. 5.4). We assume that the image to be processed is stored electronically and can be serially raster-scanned into the 2-D scrolling module. For

example, an image that is detected on a CCD chip is shifted out in a raster-scanned fashion. In the figure, the numbering in the raster-scanned image indicates the order in which the pixels are assumed to be shifted out. The image pixels are re-rastered into the 2-D scrolling module before being shifted out to the photonic signal-shifting chip. If the 2-D scrolling module is designed to be used with the CCD (analog) version of the photonic signal-shifting chip described above, then the 2-D scrolling module should itself be a CCD shift-register similar to that described in [Chiang and Chuang, 1991]. If the 2-D scrolling module and the photonic signal-shifting chip store binary-valued signals, then a thresholding stage (not shown) binarizes the raster-scanned image pixels. The 2-D scrolling module and photonic signal-shifting chip must operate at the raster-scanning clock rate. If the input images have 256×256 pixels/frame and are updated at 30 frames/sec, this implies that the 2-D scrolling module and photonic signal-shifting chip must operate at approximately 2 MHz.

Figure 5.5 shows graphically how the 2-D scrolling module functions in conjunction with the photonic signal-shifting chip in order to implement the 2-D scrolling diagrammed in Fig. 5.2 (b). First, a total of $N \times n$ clock cycles are required in order to fill the 2-D scrolling module. After another m cycles, the upper-right $n \times m$ window of the raster-scanned image is present in the photonic signal-shifting chip. With each subsequent clock cycle, the contents of the signal-shifting chip move one column to the right, which can be viewed as the $n \times m$ window shifting one column to the left in the raster-scanned image (right-most column exits the signal-shifting chip, new column enters at left-most column). After a total of $N \times n + N$ clock cycles, the window has reached the upper-left portion of the raster-scanned image. During the next $m - 1$ clock cycles the contents of

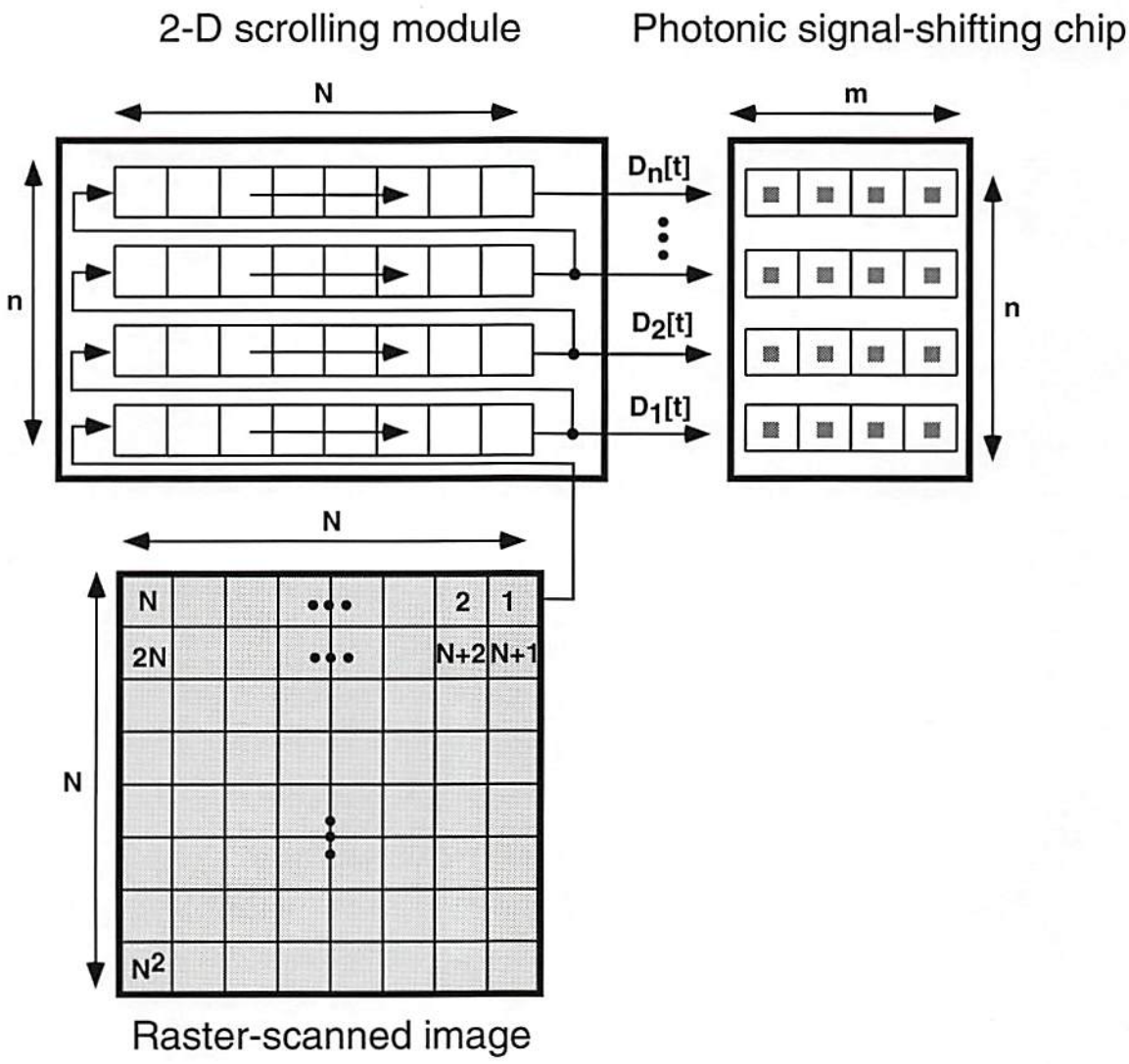


Figure 5.4: 2-D scrolling module interfaces with photonic signal-shifting chip

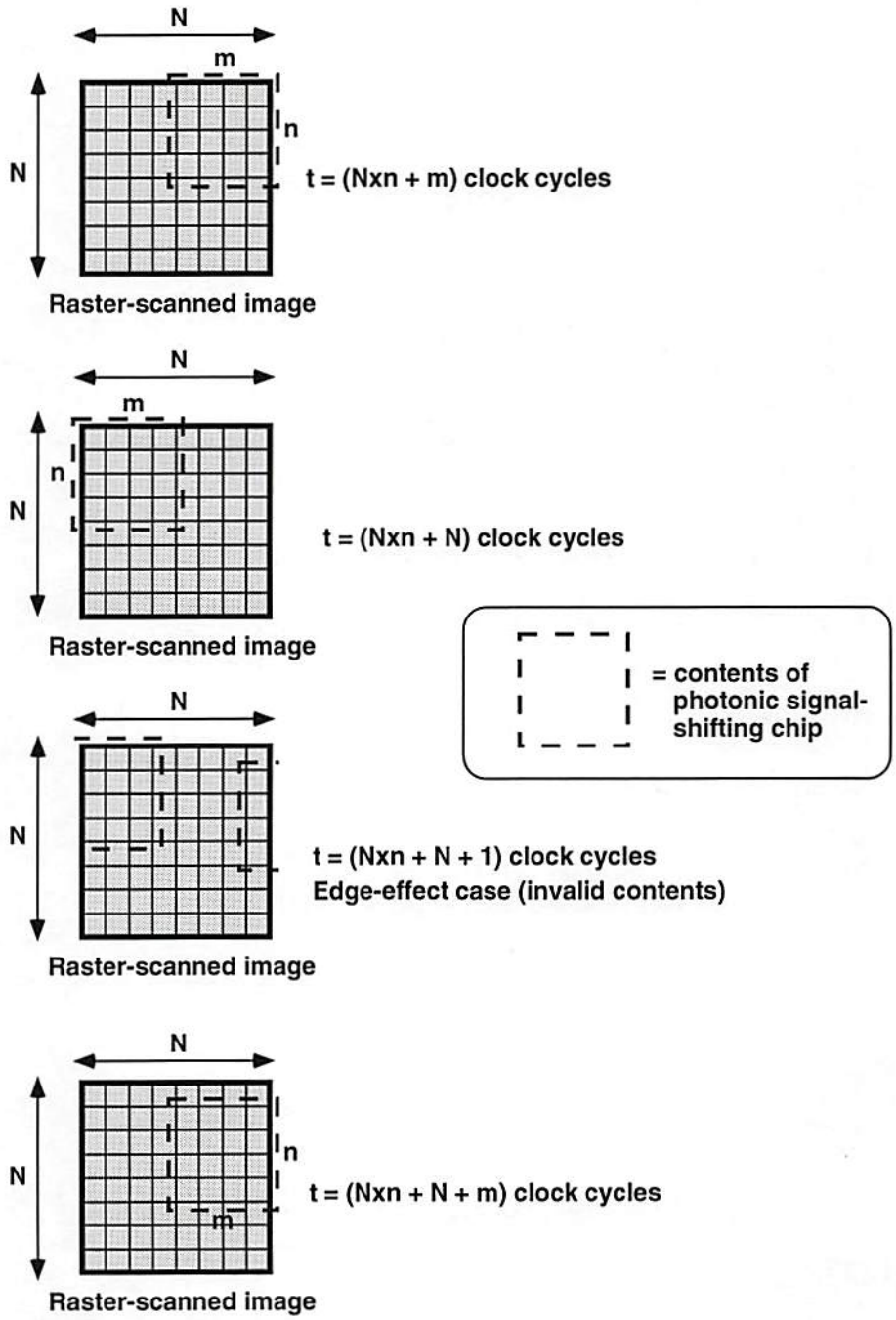


Figure 5.5: Image window contained in photonic signal-shifting chip scrolls in two dimensions. Contents of photonic signal-shifting chip are temporarily invalid while window shifts to next row.

the photonic signal-shifting chip are invalid, due to the wrap-around nature of the 2-D scrolling module. During this period, the neural network output should be ignored because the input window is corrupted. After a total of $N \times n + m$ clock cycles the contents of the photonic signal-shifting chip are once again valid. The 2-D scrolling proceeds in this fashion until all possible $(N - n + 1) \times (N - m + 1)$ valid windows have been present in the photonic signal-shifting chip.

5.4 Optical representation of neural-network signals and weights

As discussed previously, the PMCM neural-network architecture requires a dual-rail representation in order to optically represent bipolar interconnection weights. The basic dual-rail concept should be familiar to the reader from Chapter 3 and Chapter 4, in which an electronic subtraction between positive and negative detection channels takes place in the neuron units. In this section, we apply the dual-rail concept to the case of the PMCM neural-network architecture for three related purposes: the implementation of (1) bipolar interconnection weights, (2) bipolar neural input signals, and (3) optical bias cancellation. The photonic signal-shifting chip was designed with the flexibility to operate in either a single-rail mode or one of four possible dual-rail modes. These five modes of operation each have a set of associated advantages and disadvantages that will be discussed below.

5.4.1 Dual-rail optical modulators

In Fig. 5.6, the optical modulators on the photonic signal-shifting chip are grouped into dual-rail pairs, where $R_j^{(+)}$ and $R_j^{(-)}$ represent the two modulator reflectivities that optically encode the single input signal x_j. The modulators are read out using

the optical power bus described previously. The combination of the microlens and microDOE arrays results in a set of diffractive optical interconnections between each modulator and the detectors in its local fan-out region. Each neuron i in the programmable neural-network chip electronically differences its detected intensities $I_{det}^{(i+)}$ and $I_{det}^{(i-)}$, which are defined according to

$$I_{det}^{(i+)} = I_r \sum_{j=1}^{M/2} \eta_{ij}^{(+,-)} R_j^{(-)} + \eta_{ij}^{(+,+)} R_j^{(+)} \tag{5.1}$$

and

$$I_{det}^{(i-)} = I_r \sum_{j=1}^{M/2} \eta_{ij}^{(-,-)} R_j^{(-)} + \eta_{ij}^{(-,+)} R_j^{(+)}, \tag{5.2}$$

where the intensity diffraction efficiencies $\eta_{ij}^{(,)}$ are labeled as shown in Fig. 5.6, I_r is the readout intensity at each modulator, and M is the total number of modulators. (For simplicity, we have assumed that the modulators and detectors have equal areas.)

If the microDOE's are designed under the constraints that

$$\eta_{ij}^{(+,-)} = \eta_{ij}^{(-,+)} = \eta_{ij}^{(-)} \tag{5.3}$$

and

$$\eta_{ij}^{(+,+)} = \eta_{ij}^{(-,-)} = \eta_{ij}^{(+)} \tag{5.4}$$

then it is easily shown that the difference between the dual-rail detected intensities is equal to

$$I_{det}^{(i+)} - I_{det}^{(i-)} = I_r \sum_{j=1}^{M/2} (\eta_{ij}^{(+)} - \eta_{ij}^{(-)})(R_j^{(+)} - R_j^{(-)}). \tag{5.5}$$

The mapping between these dual-rail optical representations and the desired

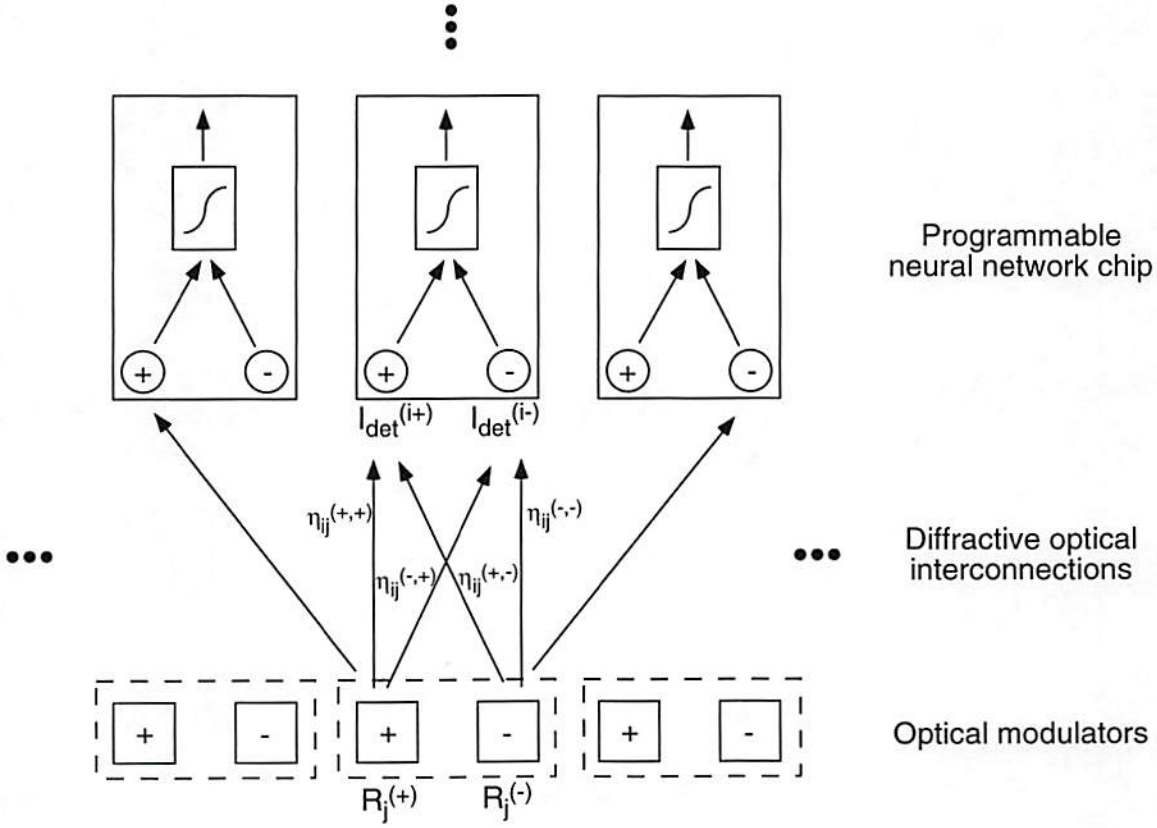


Figure 5.6: PMCM optical representation: dual-rail modulators. Paired modulator reflectivities $R_j^{(+)}$ and $R_j^{(-)}$ represent one (possibly) bipolar input signal. Dual-rail encoded diffraction efficiencies $\eta_{ij}^{(,)}$ provide bipolar neural interconnections.

neural-network computation is derived by defining the bipolar input signals x_j to be

$$x_j = (R_j^{(+)} - R_j^{(-)})/(R_{max} - R_{min}), \tag{5.6}$$

in which R_{max} and R_{min} correspond to the maximum and minimum (respectively) modulator reflectivities, and the input signals are normalized to vary between -1 and 1. Similarly, the bipolar neural-network interconnection weights can be defined by

$$w_{ij} = \eta_{ij}^{(+)} - \eta_{ij}^{(-)}, \tag{5.7}$$

where w_{ij} is the weighted interconnection between input signal x_j and neuron i. Substituting Eqs. (5.6) and (5.7) into Eq. (5.5) shows that

$$I_{det}^{(i+)} - I_{det}^{(i-)} = I_r(R_{max} - R_{min}) \sum_{j=1}^{M/2} w_{ij} x_j, \tag{5.8}$$

which verifies that the desired bipolar weighted summation takes place at each neuron. Note that this analysis assumes that the reflectivity difference $R_{max} - R_{min}$ is approximately constant over the whole modulator array.

Besides allowing for bipolar input signals, the use of a dual-rail pair of optical modulators to represent one input signal has another important advantage: the residual optical bias due to finite contrast-ratio modulation ($R_{max}/R_{min} \neq \infty$) can be canceled out. Consideration of finite contrast-ratio modulation is particularly relevant for non-resonant devices such as the SEED modulators used on our photonic signal-shifting chip. These modulators typically have a poor contrast ratio, between 2:1 and 3:1 [Krishnamoorthy *et al.*, 1996] [Hinton *et al.*, 1994].

The two finite contrast-ratio modulator reflectivities corresponding to input

signal x_j can be modeled as

$$\begin{aligned} R_j^{(+)} &= (R_{max} - R_{min})x_j^{(+)} + R_{min} \\ R_j^{(-)} &= (R_{max} - R_{min})x_j^{(-)} + R_{min}, \end{aligned} \tag{5.9}$$

in which $x_j^{(+)}$ and $x_j^{(-)}$ vary between 0 and 1, and where Eq. (5.6) implies the relation

$$x_j = x_j^{(+)} - x_j^{(-)}. \tag{5.10}$$

Eqs. (5.9) show that while neither modulator can reach a full-off state when R_{min} is non-zero, the *difference* between the two reflectivities can be zero because the R_{min} terms cancel out. As Eqs. (5.6) and (5.8) show, only the difference in reflectivities is important in terms of a neural-network computation. Thus, the effect of finite contrast ratio can be eliminated through the use of a dual-rail representation.

The photonic signal-shifting chip we fabricated was designed to allow the user to choose an appropriate signal representation. The optical modulators are driven by D flip-flops, which restricts $x_j^{(+)}$ and $x_j^{(-)}$ to being either 0 or 1. When the chip operates in what we call the *(D,0) mode*, $x_j^{(-)}$ is held at 0 and $x_j^{(+)} = x_j$ is either 0 or 1. In this mode the input signals are unipolar, but as per the above discussion, the effect of finite contrast ratio has been eliminated. When the chip operates in what we call the *(D,$\bar{D}$) mode*, the paired modulators are complementary: $x_j^{(-)} = 1 - x_j^{(+)}$. In this mode, the input signals $\{x_i\}$ are bipolar (either -1 or $+1$). The choice of whether to use *(D,0) mode* or *(D,$\bar{D}$) mode* depends on the specifics of the neural-network application and how the hidden-layer sigmoids are biased in the programmable neural-network chip.

Additional dual-rail modes

Two additional dual-rail modes are available on the photonic signal-shifting chip: *(D,D) mode* and *(D,1) mode*. These two modes are "special cases" in that the microDOE constraints given in Eqs. (5.3) and (5.4) and the bipolar signal mapping given in Eq. (5.10) do not apply.

The *(D,D) mode* can be used to give the system a degree of redundancy, in case a few isolated modulators are not functioning properly. This redundancy is obtained by driving each dual-rail pair of modulators with the same unipolar signal: $x_j^{(+)} = x_j^{(-)} = x_j$. After the photonic signal-shifting chip has been fabricated and tested, the optical power bus can be custom-designed to read out only one (functioning) modulator from each dual-rail pair.

The *(D,1) mode* can be used to provide optical bias signals to the neuron units in the programmable neural-network chip. In this mode, each dual-rail pair of modulators is driven according to $x_j^{(-)} = 1$ and $x_j^{(+)} = x_j$. The input bias signal for a given neuron unit in the programmable neural-network chip will be determined by the design of the microDOE's above the $x_j^{(-)}$ modulators. The input bias can be different for each neuron unit in the programmable neural-network chip.

5.4.2 Single-rail optical modulators

While the dual-rail representations have the advantages noted above, they also have the notable disadvantage that they require M modulators to represent $M/2$ input signals. There may be applications for which M independent input signals are needed to adequately represent the neural-network input layer. For this reason,

the photonic signal-shifting chip can also be used in a *single-rail mode.* Fig. 5.7 shows the single-rail case, in which all M modulators are fully independent. It should be noted that the single-rail configuration has twice as many independent interconnection weights as the dual-rail configuration (Fig. 5.6), as the weight constraints given in Eqs. (5.3) and (5.4) do not apply.

In deriving the mapping between the single-rail optical representations and the corresponding neural signals, a similar derivation to the one given above yields the following results:

$$I_{det}^{(i+)} - I_{det}^{(i-)} = I_r R_{max} \sum_{j=1}^{M} w_{ij} x_j, \tag{5.11}$$

in which

$$x_j = R_j / R_{max} \tag{5.12}$$

and

$$w_{ij} = \eta_{ij}^{(+)} - \eta_{ij}^{(-)}. \tag{5.13}$$

Eq. (5.12) shows the two main disadvantages inherent to the single-rail approach: the input signals are unipolar, and they are bound from below by the inverse of the contrast ratio, R_{min}/R_{max}.

5.5 VLSI design and electrical testing of the photonic signal-shifting chip

In this section, a detailed description is given for the VLSI design and electrical testing of the photonic signal-shifting chip. The chip design is first discussed at

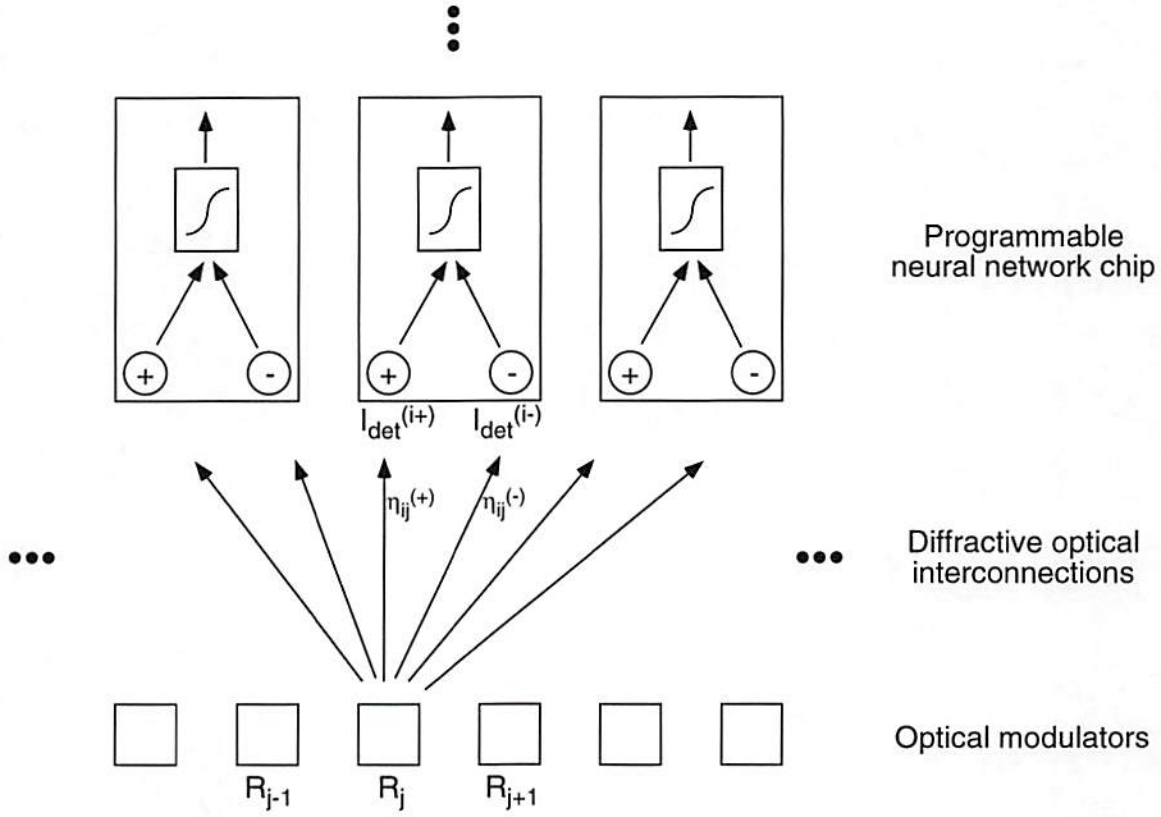


Figure 5.7: PMCM optical representation: single-rail modulators. Each modulator reflectivity R_j represents one unipolar input signal. Dual-rail encoded diffraction efficiencies $\eta_{ij}^{(+)}$ and $\eta_{ij}^{(-)}$ provide bipolar neural interconnections.

the digital gate level. The VLSI layout is then described in conjunction with photographs of the fabricated chip. Finally, simulation and electrical-test results for the photonic signal-shifting electronic chip are presented.

5.5.1 Functional description of digital CMOS circuitry

The overall layout of the photonic signal-shifting chip is shown in Fig. 5.8. The digital CMOS circuitry consists of 10 parallel shift registers, each of which contains 20 D flip-flops. This digital CMOS circuitry (0.8 μm feature size) was fabricated through the MOSIS service. Subsequently, AT&T attached a 10×20 array of GaAs SEED modulators to the CMOS chip using their flip-chip bonding and substrate-removal processes [Goossen *et al.*, 1995]. Each of the 200 SEED modulators is driven by one of the CMOS D flip-flops.

The photonic signal-shifting chip consists of two repeated circuit structures: "dual-rail signal generators" and "basic cells" (Fig. 5.9). Each shift register consists of a dual-rail signal generator and 10 basic cells, where a basic cell consists of a pair of D flip-flops that drive adjacent modulators. Three control lines configure the dual-rail signal generators and basic cells to operate in one of the five possible optical representation modes described previously.

The dual-rail signal generation circuitry is used to convert a single input signal into a dual-rail pair of signals, as determined by the control lines S0 and S1. Figure 5.10 shows that each input signal Din is converted into a pair of output signals D1 and D2, such that D1=Din and D2 is one of 4 possible Boolean functions of Din. For example, the control values S1=0,S0=1 select the *(D,$\bar{D}$) mode*, where output signal D2 is the complement of output signal D1. The 4 boolean functions are implemented using an exclusive-or gate and a 2-to-1 multiplexer.

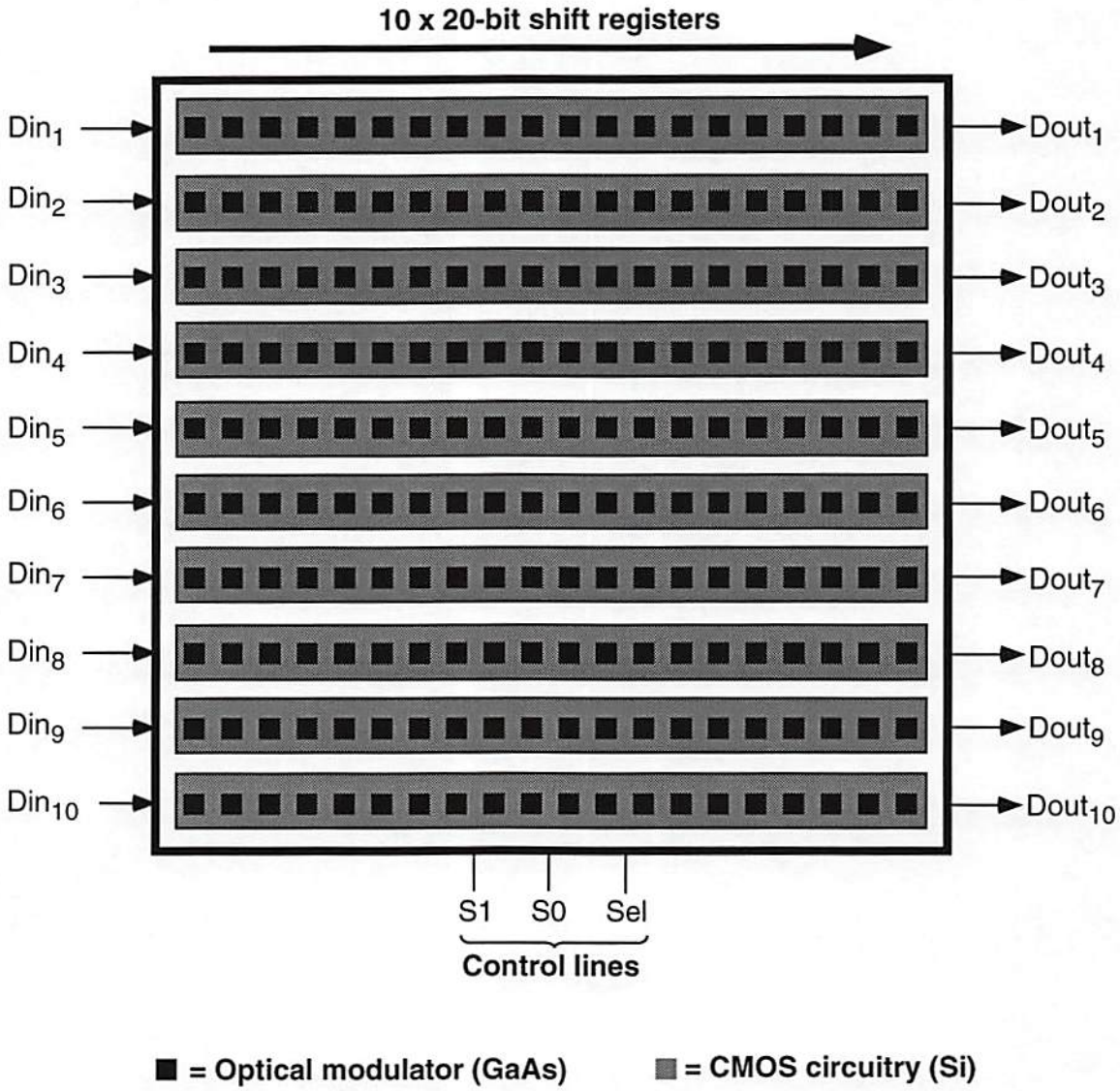


Figure 5.8: Overall layout of the photonic signal-shifting chip

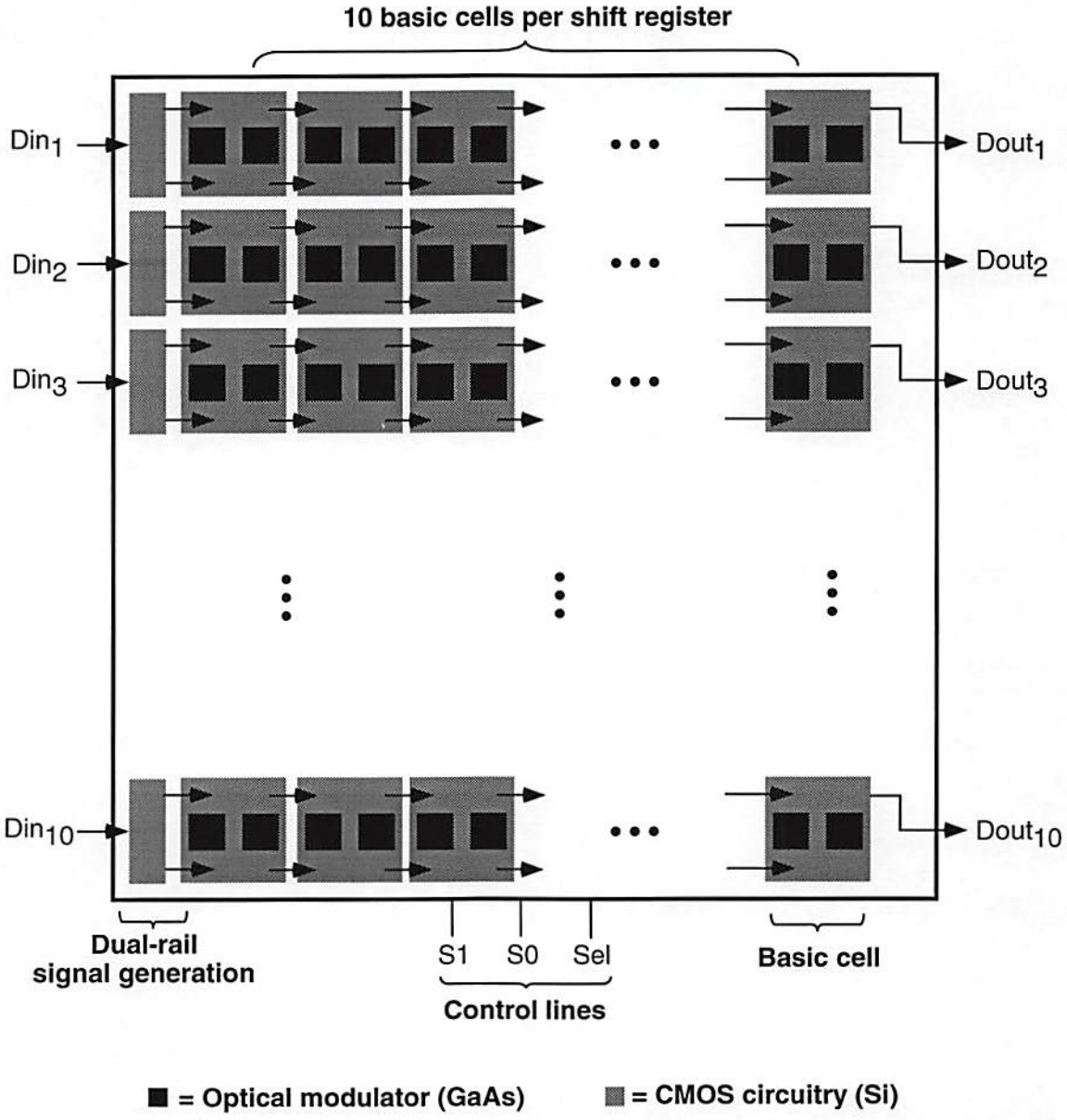
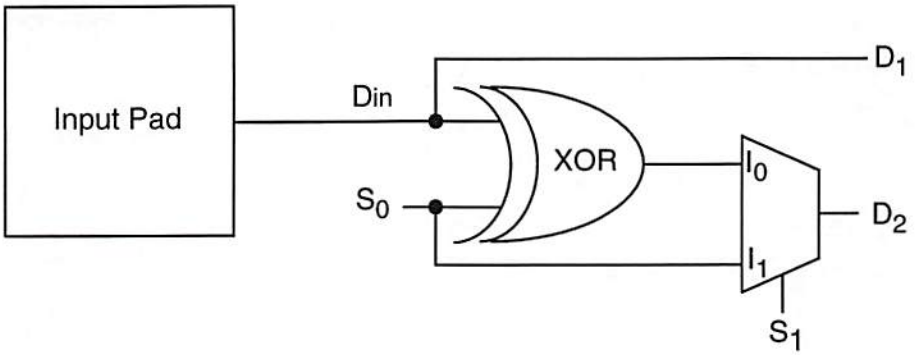


Figure 5.9: Circuit structures in the photonic signal-shifting chip: basic cells and dual-rail signal generation



S_1	S_0	D_1	D_2	Mode
0	0	Din	Din	(D,D)
0	1	Din	$\overline{\text{Din}}$	(D,$\overline{\text{D}}$)
1	0	Din	0	(D,0)
1	1	Din	1	(D,1)

Figure 5.10: Dual-rail signal-generation circuitry. Control signals S1 and S0 select which of the 4 dual-rail representation modes are used.

The basic-cell circuitry is shown in Fig. 5.11. The cells are arranged such that inputs D1 and D2 are connected to outputs Q1 and Q2 from the previous cell (respectively), except that the first basic cell in each row is connected to the dual-rail signal generator. Internally, each basic cell contains two D flip-flops FF1 and FF2 that drive SEED modulators M1 and M2 (respectively). The control signal Sel determines how the binary signals flow through the cell. When Sel=1 the dual-rail mode of operation in selected, in which a pair of bits is shifted into the cell and a pair of bits is shifted out from the cell in one clock cycle. When Sel=0 the single-rail mode of operation is selected, in which the binary signals follow a 2 clock-cycle serpentine path through the cell: D1 $\rightarrow$ FF1 $\rightarrow$ FF2 $\rightarrow$ Q1. A complete shift register is formed between an input Din and output Dout by connecting 10 basic cells in series (as shown in Fig. 5.9). There is a delay of 10

clock cycles between Din and Dout when Sel=1; when Sel=0 there is a 20 clock cycle delay between Din and Dout.

5.5.2 VLSI layout and chip photographs

The VLSI layout for the photonic signal-shifting chip was constructed from a set of 3-metal-layer scalable-CMOS standard cells supplied by AT&T, MOSIS, and Tanner Research. The layout is shown in Fig. 5.12. It consists of approximately 12,000 transistors and occupies a 1.95 μm $\times$ 1.95 μm area using a 0.8 μm feature size. The SEED modulators are electrically contacted through 20 μm $\times$ 20 μm solder-bump bonds deposited by AT&T, and we accessed the solder-bump bonds using metal-3 (third layer metal) lines. The chip was designed to operate at a clock frequency of at least 2 MHz (the raster-scan rate for a 256 $\times$ 256 image updated at 30 frames/sec).

Figure 5.13 shows photographs of the fabricated chip taken under a microscope. In Fig. 5.13 (a), the complete 10×20 array of modulators is visible in the interior portion of the chip. The 10 $\times$ 20 array of modulators has a pitch of 62.5 μm in the horizontal dimension and 125 μm in the vertical dimension. Figure 5.13 (b) is a closeup of one basic cell that drives a pair of modulators. The AT&T substrate removal process allows visual access to both the CMOS circuitry and the GaAs SEED modulators.

5.5.3 Simulation and electrical-test results

One of the 10 identical shift registers was extracted from the VLSI layout using the ":extract" command in the MAGIC layout editor. The extracted ".ext" file

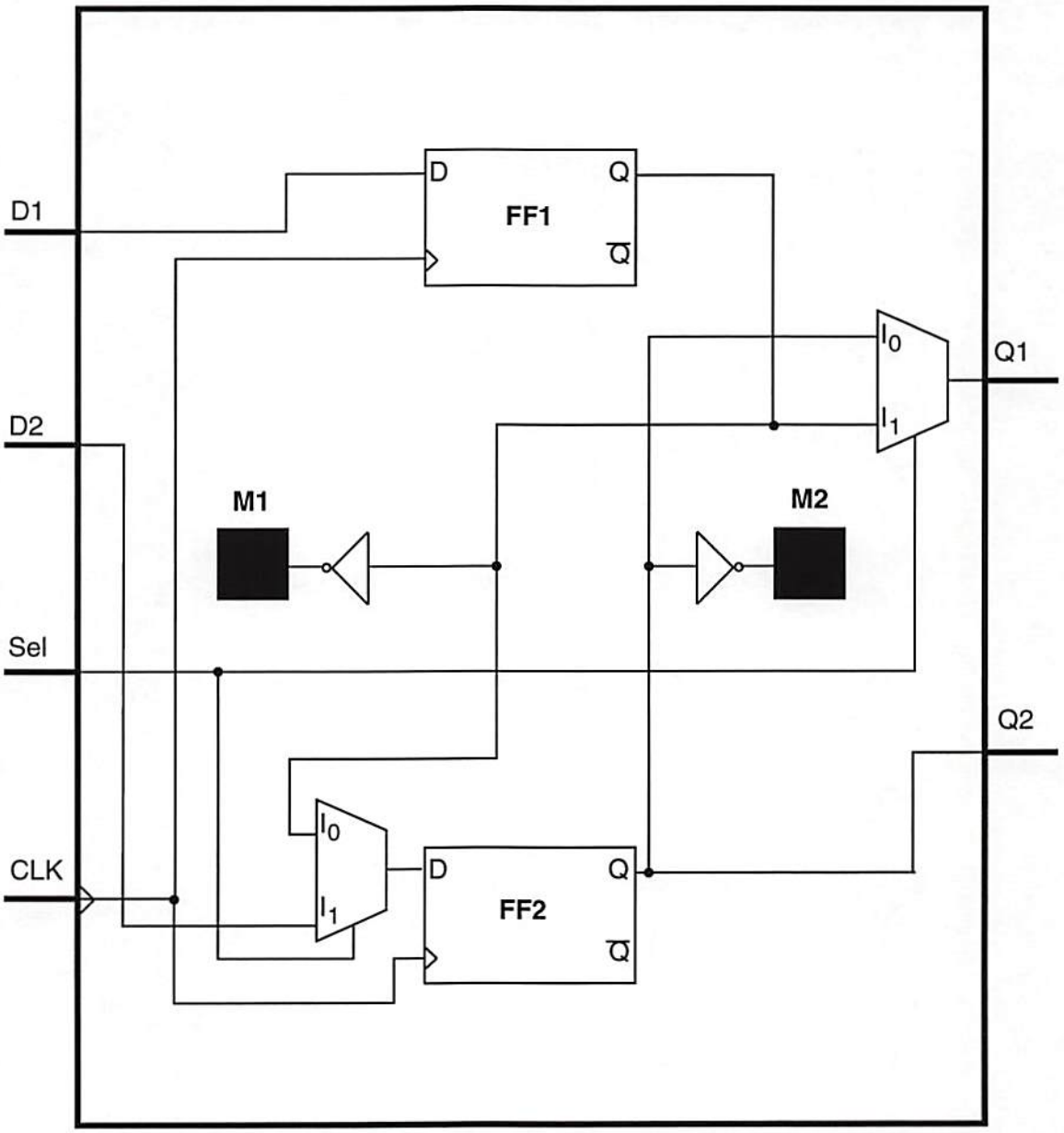


Figure 5.11: Basic-cell circuitry. When Sel=1 (dual-rail mode selected), at each clock cycle a pair of bits is shifted into the cell (from D1 and D2) and a pair of bits is shifted out to the next cell (through Q1 and Q2). When Sel=0 (single-rail selected), at each clock cycle 1 bit is shifted in from D1 to FF1, 1 bit is shifted from FF1 to FF2, and 1 bit is shifted from FF2 to the next cell through output Q1 (D2 and Q2 are ignored). The flip-flops FF1 and FF2 drive SEED modulators M1 and M2.

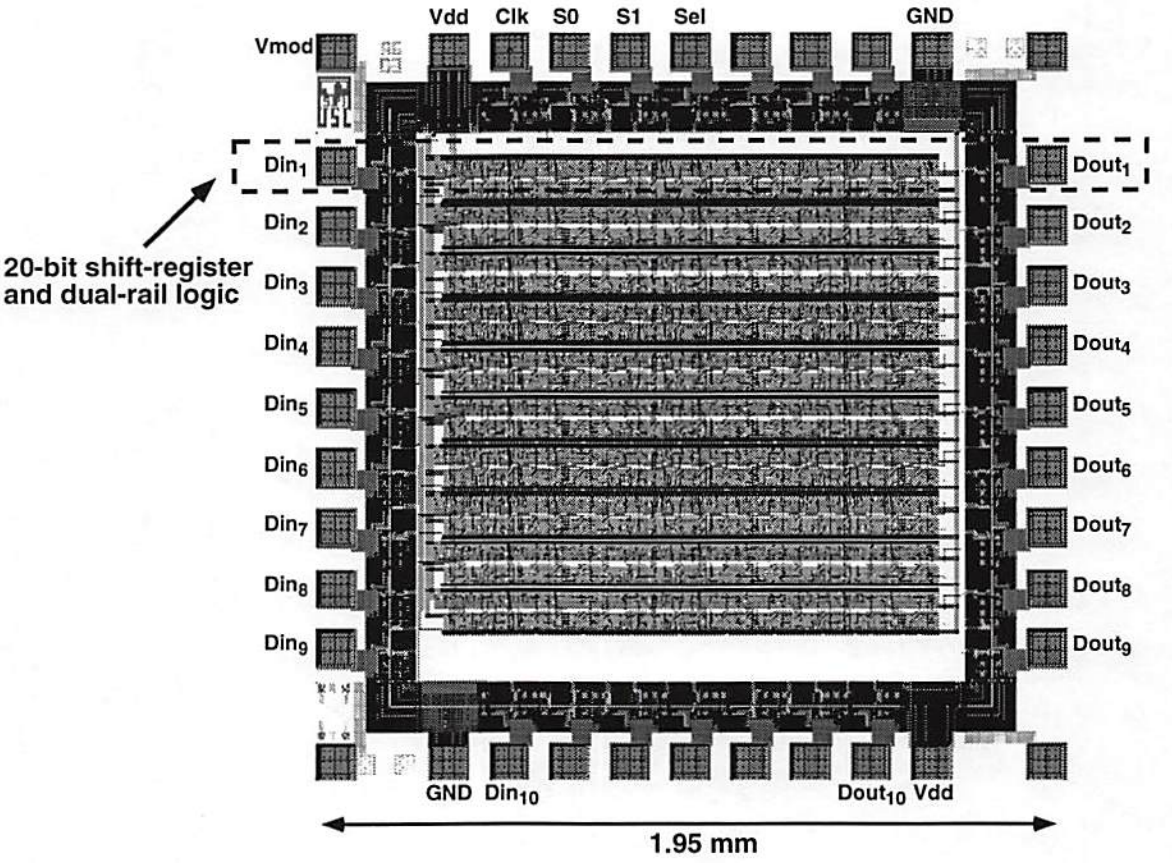
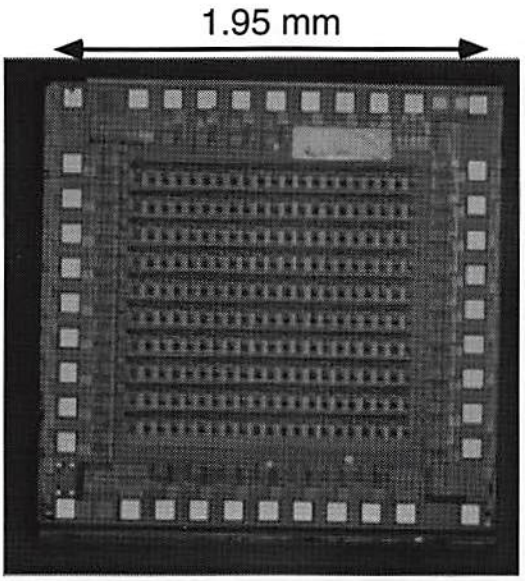
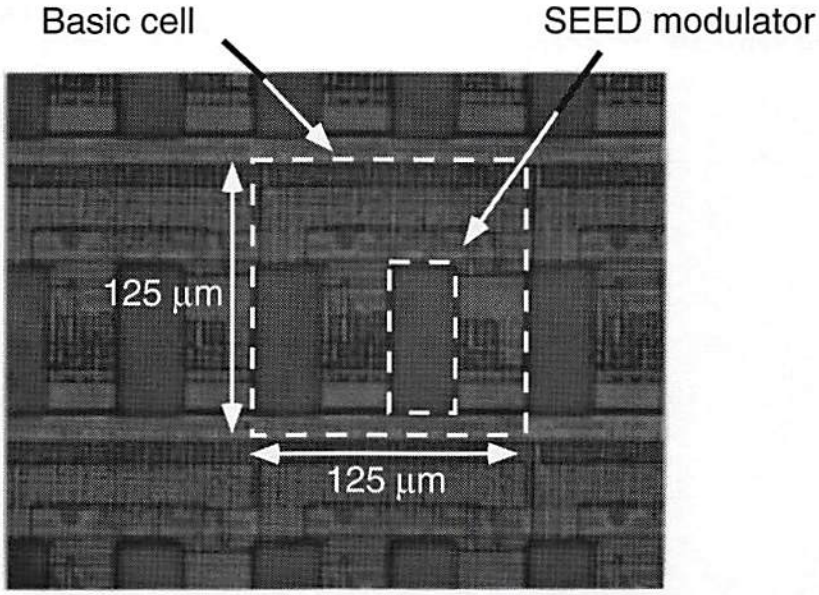


Figure 5.12: Photonic signal-shifting chip VLSI layout



(a)



(b)

Figure 5.13: Photonic signal-shifting chip photographs; (a) whole chip (b) closeup of a basic cell.

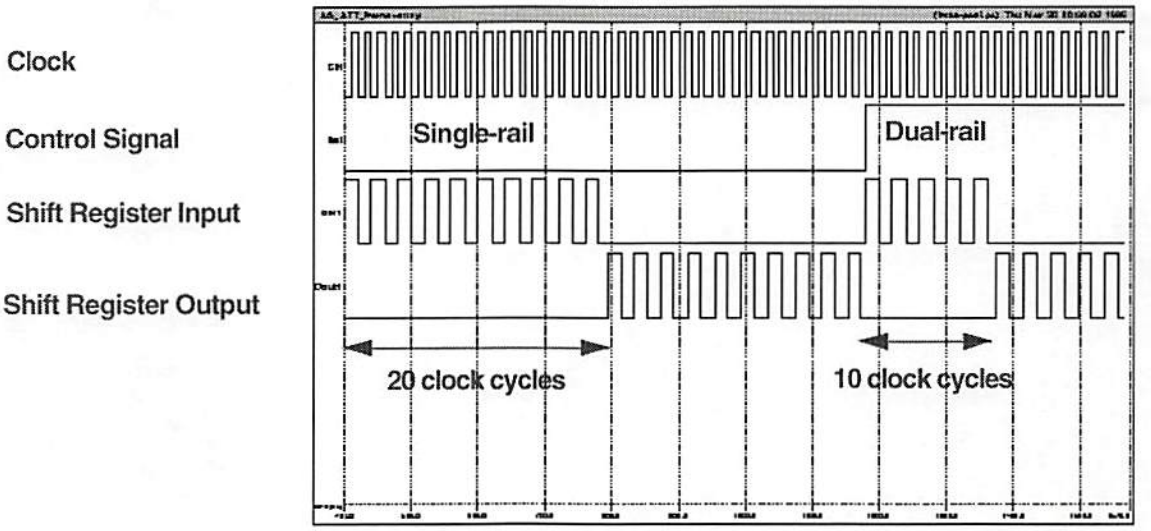


Figure 5.14: Switch-level circuit simulation results (irsim). In single-rail mode (Sel=0), there is a 20-bit delay between the shift-register input and shift-register output signals. In dual-rail mode (Sel=1), there is a 10-bit delay between the shift-register input and shift-register output signals.

was then converted to a ".sim" file (using the "ext2sim" command on our UNIX system), which was then used as the input file for the switch-level simulator, "irsim". The simulation results are shown in Fig. 5.14. Initially, all 20 flip-flops were reset to 0 and the single-rail mode was selected by setting the dual-rail select line to Sel=0 (see Fig. 5.11). An oscillating binary signal (1 MHz) was then input to the first flip-flop in the shift register. As shown in the figure, there was a 20 clock-cycle delay between the shift-register input and output signals, which indicates that the single-rail mode was functioning correctly. After all 20 flip-flops were cleared, a dual-rail mode was selected by setting the dual-rail select line to Sel=1. (The settings on the S1 and S0 control lines were irrelevant for these simulations.) As shown in the figure, a 10 clock-cycle delay occurred between the shift-register input and output signals, which indicates that the dual-rail mode was functioning correctly.

AT&T delivered a packaged electrical-only version of our chip for testing purposes. The first test was to verify that none of the 200 D flip-flops was permanently stuck in either the $Q = 0$ state or the $Q = 1$ state. This was verified by operating the chip in single-rail mode and daisy-chaining the output of each 20-bit shift register to the input of the next 20-bit shift register (*i.e.*, $Dout_1 \rightarrow Din_2$, $Dout_2 \rightarrow Din_3$, *etc.*). In this way, all 200 D flip-flops were connected into one long shift register, and if any one of the flip-flops were stuck at $Q = 0$ or $Q = 1$ then the input signal could not reach the output of the final flip-flop. The oscilloscope traces from the first test are shown in Fig. 5.15. A 2 MHz clock signal is shown in the upper trace and the output signal from the last flip-flop is shown in the lower trace. The output signal oscillates at the same frequency as the 1 MHz input signal (the input is not shown), which indicates that all 200 flip-flops are operational.

The next test was to verify that each shift register functions as a 10-bit delay line when operating in a dual-rail mode and that it functions as a 20-bit delay line when operating in the single-rail mode. Figures 5.16 and 5.17 verify that both modes function correctly. The 2 MHz clock signal is shown in the upper traces, and the output from one of the ten shift-registers ($Dout_1$) is shown in the lower traces. Initially, all flip-flops were reset to 0. In the experiment shown in Fig. 5.16, 10 clock cycles elapse before the output signal starts to oscillate, thereby verifying that the dual-rail mode is functioning correctly. Figure 5.17 shows the single-rail case, in which 20 clock cycles elapse before the signal starts to oscillate.

The above tests indicate that the CMOS electronics on the photonic signal-shifting chip is functioning correctly. Optical testing of the 2-D array of SEED modulators has not yet been completed, however. These optical tests will be left

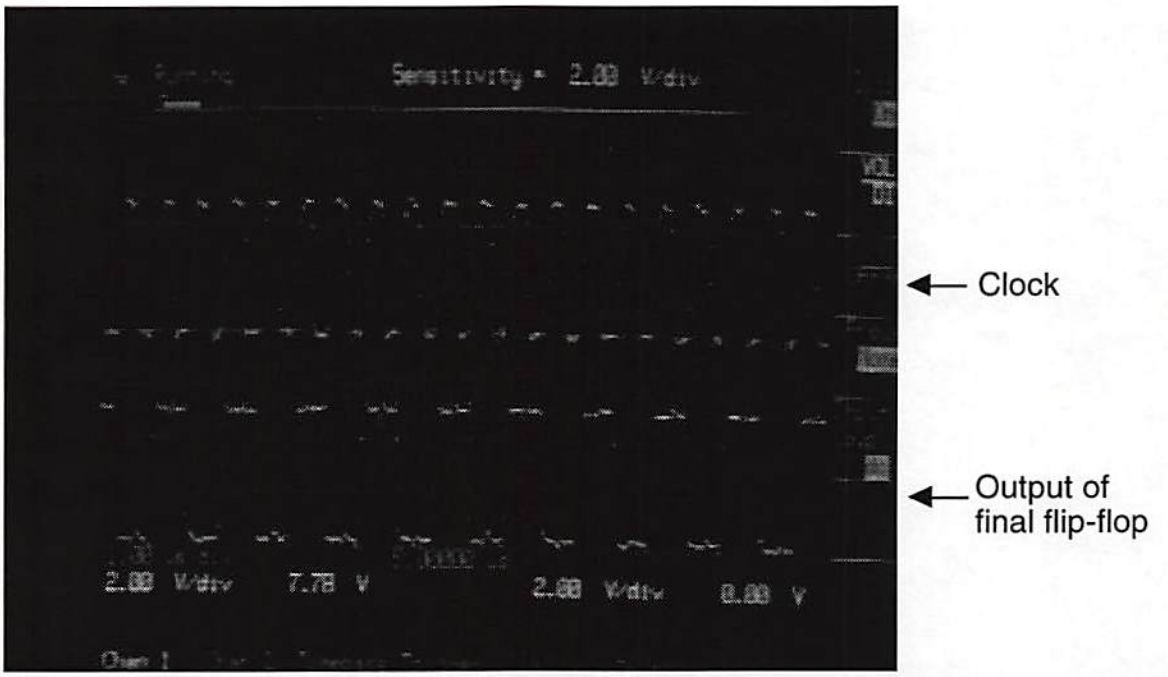


Figure 5.15: Electrical test of all flip-flops in the photonic signal-shifting chip. All 200 D flip-flops are daisy-chained together into one long shift register. Upper oscilloscope trace is a 2 MHz clock signal. Lower trace is the output signal from the final flip-flop (the input signal is not shown).

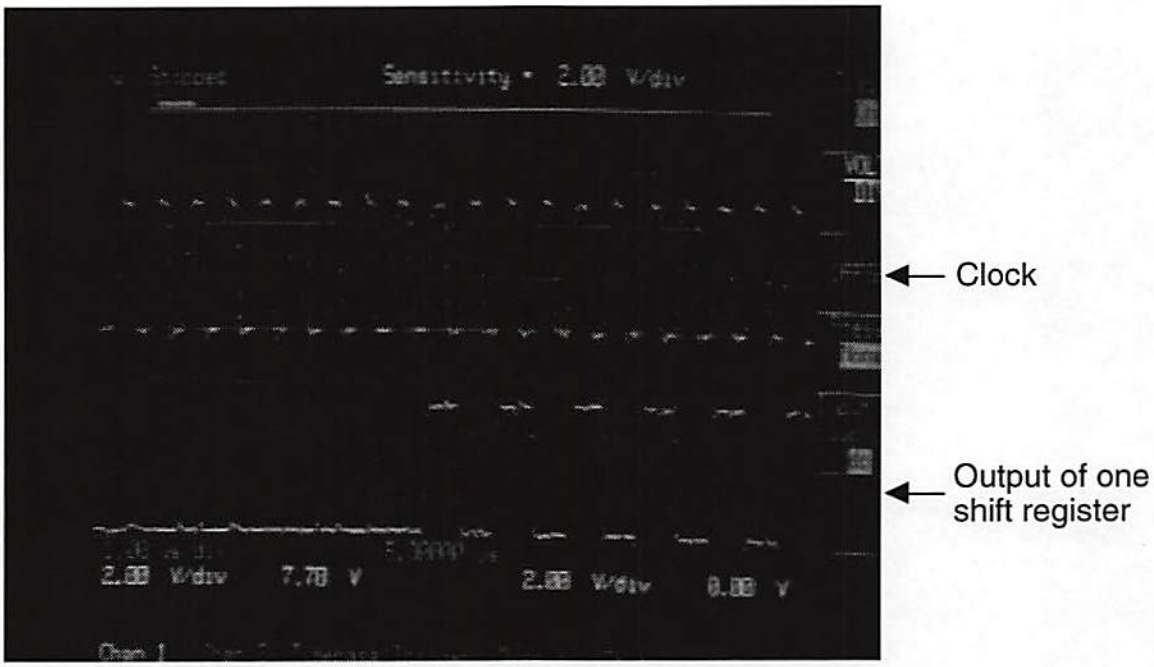


Figure 5.16: Electrical test of the photonic signal-shifting chip operating in a dual-rail mode. Upper oscilloscope trace is a 2 MHz clock signal, lower trace is the output signal from one shift register. All flip-flops were initially reset to 0. The oscilloscope was triggered off the first rising edge of the 1 MHz input signal oscillation (not shown). There is a 10 clock-cycle delay between the input signal oscillation and the output signal oscillation

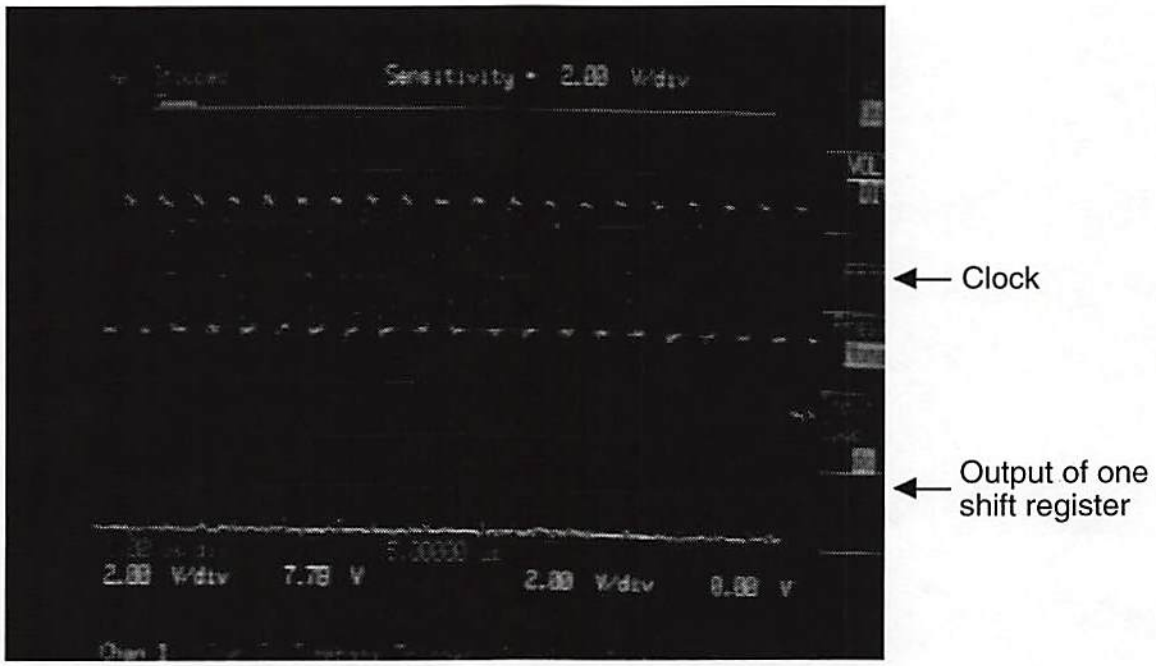


Figure 5.17: Electrical test of the photonic signal-shifting chip operating in single-rail mode. Upper oscilloscope trace is a 2 MHz clock signal, lower trace is the output signal from one shift register. All flip-flops were initially reset to 0. The oscilloscope was triggered off the first rising edge of the 1 MHz input signal oscillation (not shown). There is a 20 clock-cycle delay between the input signal oscillation and the output signal oscillation

for a future investigation.

5.6 Design and fabrication of space-variant microDOE's

In this section, the focus of the discussion shifts to the optical interconnection elements that were shown in Fig. 5.1. Once the input signals are converted into optical signals by the photonic signal-shifting chip, the function of the microDOE and microlens arrays is to optically compute the weighted fan-out and fan-in required by the first layer of neural-network interconnections. We designed several such microDOE and microlens arrays (as a collaborative project described below) and had them fabricated by Honeywell. In the discussion below, we first describe the neural-network computations implemented by the microDOE array at the systems level, and then go on to discuss some technical issues that arose during the design process. The design parameters for an optical system composed of a space-variant microDOE array and a corresponding microlens array are then derived, in order to show the feasibility of constructing a compact 3-D PMCM optical interconnection system using currently available technology.

5.6.1 Computational complexity and space-variant microDOE's

A microDOE array interfaces with the photonic signal-shifting chip as shown in Fig. 5.18 (drawn in 1-D for convenience). As discussed previously, each microDOE causes a modulated input beam to be split into a set of weighted fan-out beams. The optical superposition of the fan-out beams from nearby microDOE's

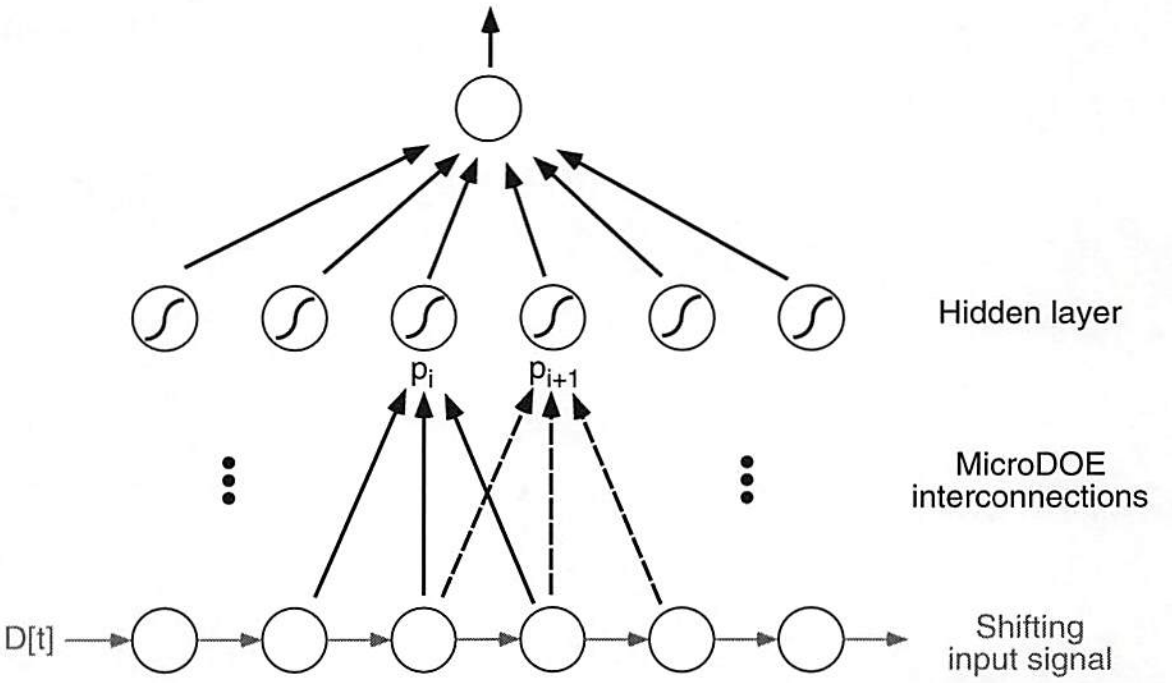


Figure 5.18: MicroDOE interconnections and a shifting input signal. If the interconnections are space-invariant, then the neural input potentials are related according to $p_{i+1}[t+1] = p_i[t]$, where t denotes a discrete time step. If the interconnections are space-variant, then all potentials must be recomputed at each time step.

results in one layer of local neural-network interconnections. If each of n microDOE's fans-out to F hidden-layer neurons, then $n \times F$ multiplications and $n \times F$ additions are computed at each clock cycle. As these computations are analog, they should be considered arithmetic operations (low precision) rather than floating-point operations. If the photonic signal-shifting chip operates at f clock cycles per second, then the diffractive optical elements can compute $2 \times n \times F \times f$ arithmetic operations per second. Using numbers appropriate for the hardware we have fabricated to date ($n = 10 \times 20$, $F = 3 \times 3$, $f = 2$ MHz), a very small-scale (2 mm $\times 2$ mm in cross-section) system can compute 7.2×10^9 arithmetic operations per second.

As indicated in Fig. 5.18, the space variance of the microDOE array is crucial to the above computational complexity analysis. If the fan-out weights from each microDOE are identical, meaning that the design is space invariant, then the fan-in weights entering each neuron are also identical. (Technically, true space invariance is impossible because the fan-out weights at the left and right edges must be different than the internal fan-out weights. This "edge effect" can be ignored for a sufficiently large input layer.) In this case, each neural input potential (weighted summation) $p_i[t]$ calculated by hidden-layer neuron i at time-step t is identical to the potential $p_{i+1}[t+1]$ calculated by hidden-layer neuron $i+1$ at time-step $t+1$. Therefore, almost all of the $2 \times n \times F$ calculations performed by the microDOE array are redundant. Disregarding the "edge effect" described above, only the $2 \times F$ multiplications and additions that compute the left-most (internal) input potential are required at each clock cycle. This reduces the computational complexity from 7.2×10^9 down to 36×10^6 arithmetic operations per second using the parameters given above. As a result, we primarily designed space-variant microDOE arrays for fabrication by Honeywell.

5.6.2 Space-variant microDOE-array design process

The process of designing the space-variant microDOE arrays involved several computer processing steps. This was a collaborative effort, in which the tasks were divided as follows:

1. *Specification of neural-network interconnection weights.* These weights can either be designed by hand to transform the input layer into a feature-extracted hidden-layer, or the weights can be trained for a specific application using a neural-network learning algorithm such as backpropagation. In

either case, the weights must conform to the local fan-out constraints imposed by the microDOE-array hardware. The microDOE's discussed in this dissertation were specified (by the author) using the feature extraction approach. The weights implement "space variant edge detection" operations, which are shown to be useful neural-network features in Chapter 6.

2. *Design of periodic phase-only patterns that yield diffraction efficiencies proportional to the neural-network interconnection weights.* This work was done by Ching-Chu Huang (from Dr. Jenkins' research group at USC), who used a variant of the Gerchberg-Saxton algorithm (as described in [Huang, *et al.*, 1997]) to design the space-variant microDOE arrays. These phase-only patterns were quantized to 8 uniformly-spaced phase levels, in accordance with the DOE fabrication process available from Honeywell.

3. *Conversion of phase-only patterns into mask definition files.* The 8 uniformly-spaced phase levels were etched using 3 fabrication masks, for phase lags of π, $\pi/2$, and $\pi/4$ radians (see [Jahns, 1994] for more fabrication details). Dr. Charles Kuznia wrote the software that converts the phase-only patterns into 3 "CIF" (Caltech Intermediate Form) mask definition files.

4. *Data compression using polygon descriptors.* It was found that the original CIF data files were prohibitively large because they were defined by minimum feature-sized square blocks. For example, Fig. 5.19 (a) shows one period of a DOE mask that is described by a collection of individual square features. We found that the file sizes could be substantially reduced by merging the individual square features into a minimum number of equivalent polygons, as shown in Fig. 5.19 (b). Although some VLSI layout tools

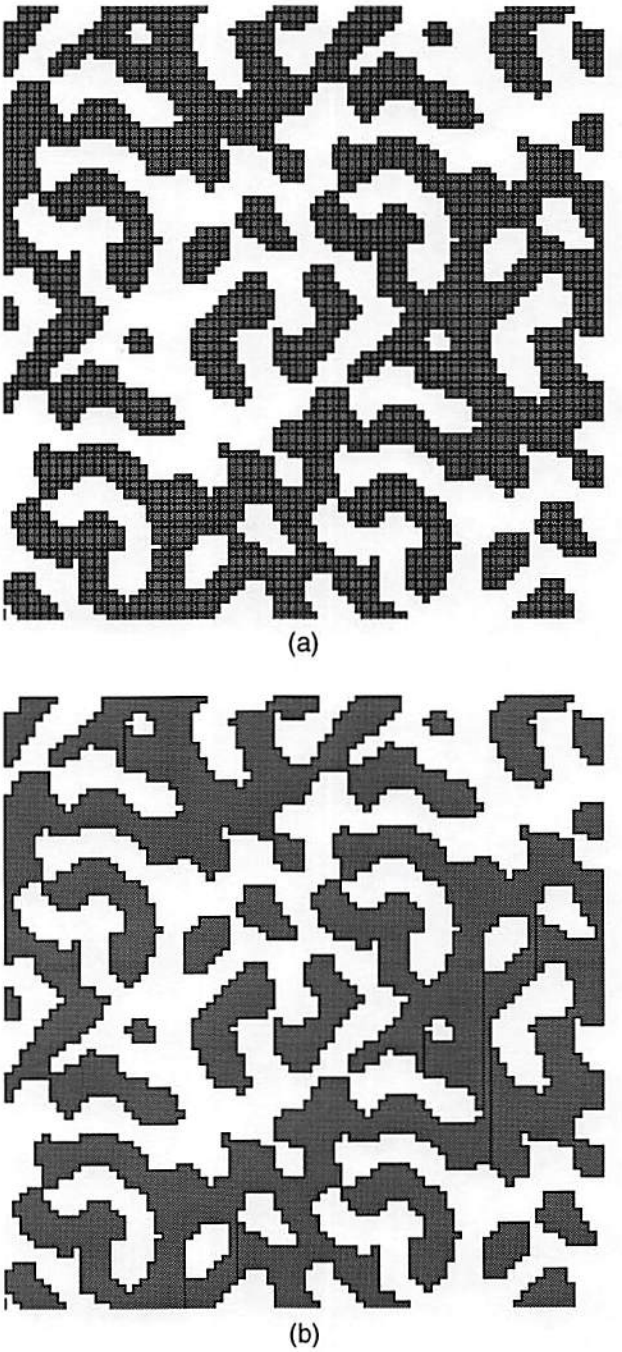


Figure 5.19: Data compression using polygon descriptors. In (a), one period of a DOE is represented as a large number of minimum-feature sized blocks. Our custom software merges the individual features into a minimum number of equivalent polygons, shown in (b). The file size is reduced by a factor of 12 (in GDSII format) for the example shown.

(such as L-Edit) can perform this merging operation, we found that custom software was needed to rapidly process the hundreds of microDOE's that had been designed for our space-variant neural-network interconnections. The program "polygonize" (written by the author) rapidly compresses CIF data files using a simple polygon tracing algorithm. The program resides on our UNIX system at USC SIPI. Compression of the data files is very important from a practical point of view, in that mask fabrication costs scale with the size of the data files.

5.6.3 3-D PMCM optical system design

The microDOE arrays were designed to operate in a 3-D PMCM optical system such as that shown in Fig. 5.20. As shown in the figure, the microDOE array interconnects each modulator on the photonic signal-shifting chip to a local set of detectors on the programmable neural-network chip. The microlens array images the modulator plane onto the detector plane. Under idealized assumptions (point-source modulators, no aberrations, infinite lens aperture), each microDOE diffracts the incident light into point sources on the detector plane in proportion to the Fourier transform of its phase-only transmittance function [Goodman, 1996]. The transmittance function of each microDOE is thus designed to yield a Fourier transform consistent with the desired neural-network interconnection weights. To show the feasibility of constructing such an optical system in a compact volume, we calculate below the propagation distances and diffraction-limited spot sizes of a 3-D PMCM optical system, where the finite extents of the modulators, microDOE's, and microlenses are taken into account.

The optical interconnection system parameters were determined using four

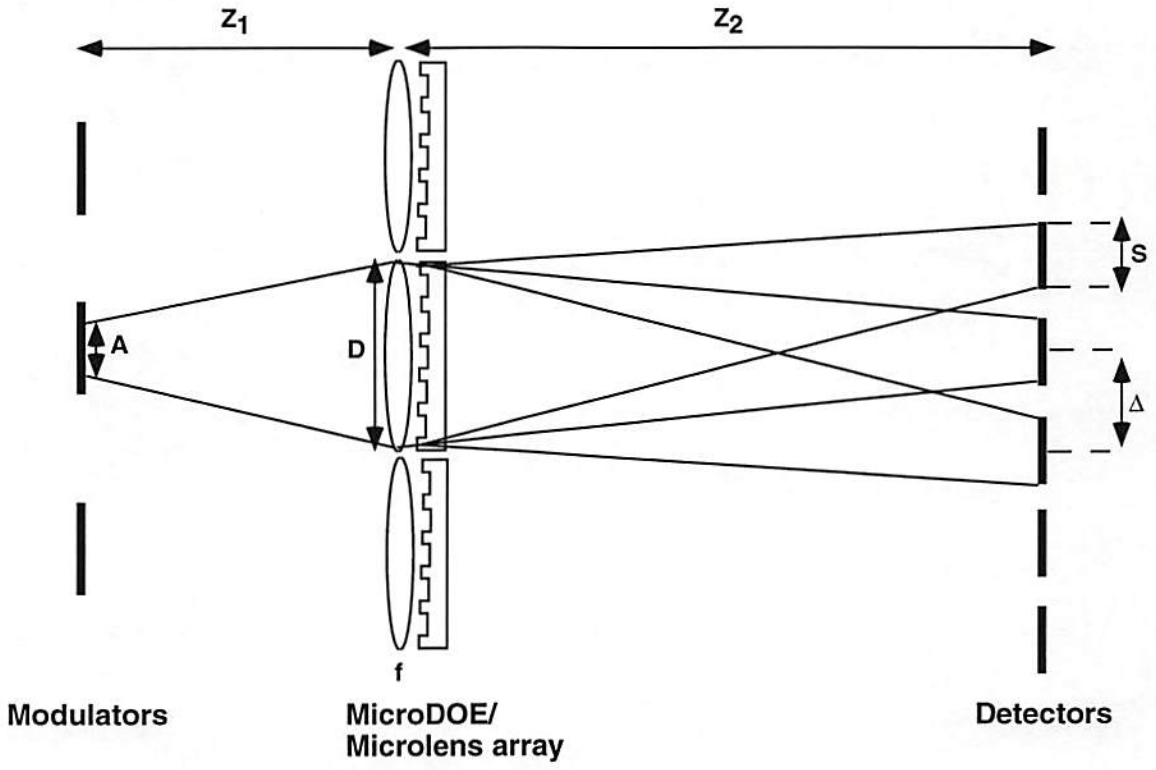


Figure 5.20: 3-D PMCM optical system parameters

design constraints:

1. *Imaging condition.* To satisfy the optical imaging condition between the modulator and detector planes, propagation distances Z_1 and Z_2 are related to the microlens focal length f in accordance with [Goodman, 1996]

$$\frac{1}{Z_1} + \frac{1}{Z_2} = \frac{1}{f}. \tag{5.14}$$

2. *Illumination of one microDOE/microlens.* We assume that the optical power bus (not shown in the figure) outputs a uniform-phase and uniform-amplitude beam with (1-D) spatial extent A. This uniform beam immediately reflects off of a modulator and expands, due to diffraction, into a beam with a (1-D) spatial extent equal to D, where D represents the microDOE/microlens aperture. If the microDOE/microlens plane is in the Fraunhofer regime with respect to the modulator plane (which will be subsequently verified), then

$$D = 2\lambda Z_1 / A \tag{5.15}$$

must be satisfied in order for the central lobe of the Fraunhofer diffraction pattern to illuminate one microlens and one microDOE [Goodman, 1996].

3. *Spot spacing equals detector spacing.* The spacing Δ between diffracted spots is related to the microDOE period length T, propagation distance Z_2, and wavelength λ through the relation [Huang, *et al.*, 1997]

$$\Delta = \lambda Z_2 / T. \tag{5.16}$$

The propagation distance and period length are chosen such that Δ is equal to the spacing between detectors on the programmable neural-network chip.

4. *Spot size less than spot spacing.* The shape of the detected spots can be modeled as the convolution of the geometric-optical image of the illumination source with the Fourier transform of the imaging-system exit pupil [Goodman, 1996]. For simplicity, we assume a diffraction-limited imaging system, which implies a uniform pupil function with an extent equal to the microlens/microDOE aperture D. Thus, the intensity of the imaged spot can be modeled (in 1-D) as

$$I(x) \propto \left|\mathrm{rect}[x/(MA)] * \mathrm{sinc}[xD/(\lambda Z_2)]\right|^2, \tag{5.17}$$

where x is a spatial variable evaluated at the detector plane, $\mathrm{rect}(x) \triangleq 1$ for $|x| < (1/2)$ and 0 otherwise, $\mathrm{sinc}(x) \triangleq \sin(\pi x)/(\pi x)$, and $M = Z_2/Z_1$ is the optical magnification between the modulator and detector planes. The spot size S, which can be taken as the central-lobe width of $I(x)$, must be smaller than the spot spacing Δ to avoid high levels of crosstalk. Preferably, $S < \Delta/2$.

In order to image each of the 200 photonic signal-shifting chip modulators onto the detector array, the microDOE/microlens apertures can be no larger than the 62.5 μm modulator pitch. It was found that with only 62.5 μm $\times$ 62.5 μm of microDOE area, it was necessary to use a 1.5 μm minimum feature size in order to achieve the space-bandwidth product required by the neural-network interconnection weights. A feature size of 1.5 μm may be pushing the limits of current DOE fabrication technology, and it is also approaching the optical

wavelength $\lambda = .85\ \mu m$, which implies that complicated vector-diffraction theory may be required for reasonably accurate microDOE design [Goodman, 1996]. As a result, the optical system described below uses a $125\ \mu m \times 125\ \mu m$ microDOE and microlens aperture with a minimum feature size of $2.5\ \mu m$. In so doing, half of the modulators on the photonic signal-shifting chip are unused in the optical system, resulting in a 10×10 array of usable modulators rather than the full 20×10 array.

The optical system parameters for one of our microDOE array (identification code: D9504Hf) and corresponding microlens array (identification code: L9504Bf) designs are given as follows: $D = 125\ \mu m$, $Z_1 = 1103\ \mu m$, $Z_2 = 1470\ \mu m$, $M = 1.33$, $f = 630\ \mu m$, $A = 15\ \mu m$, $\lambda = .85\ \mu m$, $T = 20\ \mu m$, and $\Delta = 62.5\ \mu m$. These parameters are consistent with Eqs. (5.14)-(5.16) given above. The Fraunhofer approximation implicit to Eq. (5.15) is checked by verifying that the Fraunhofer condition $Z_1 > (2A^2)/\lambda$ [Goodman, 1996] is satisfied in this case. The spot size calculation given in Eq.(5.17) is numerically evaluated in Fig. 5.21 using the parameters given above. The figure indicates a spot size of $S \approx 30\ \mu m$, which approximately satisfies the condition $S < \Delta/2$.

A portion of the fabricated 10×10 array of microDOE's designed using the above parameters is shown in Fig. 5.22. The microDOE's were designed to provide weighted fan-out to a 3×3 nearest-neighbor set of neurons in the programmable neural-network chip, which corresponds to a 6×3 physical fan-out due to the dual-rail encoding of the bipolar interconnection weights. The figure shows a photograph of the glass (SiO_2) microDOE elements. A portion of a corresponding 10×10 array of $125\ \mu m \times 125\ \mu m$ microlenses is shown in Fig. 5.23. These microlenses were designed (by Jen-Ming Wu, from Dr. Sawchuk's research group)

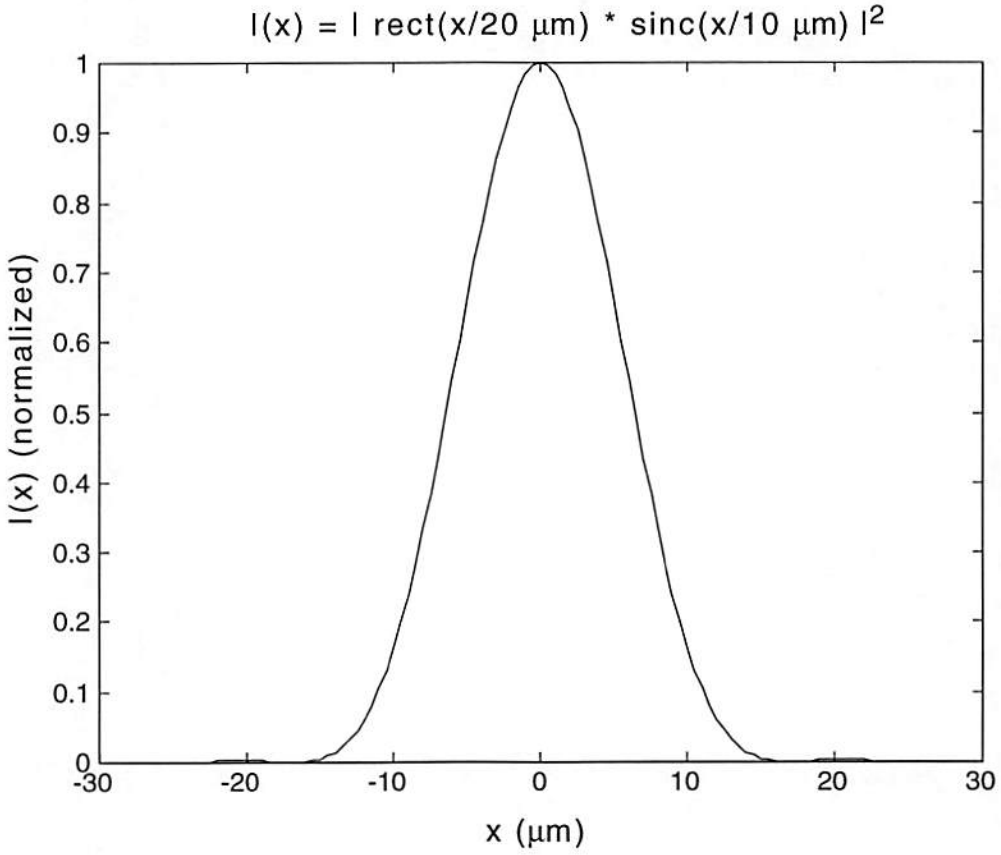


Figure 5.21: Spot size calculation

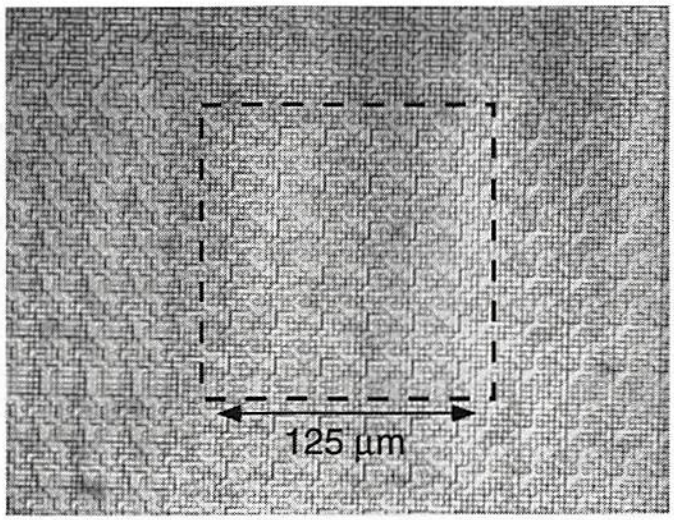


Figure 5.22: A photographed portion of a 10×10 microDOE array (identification code: D9504Hf). Each microDOE is 125 μm on a side.

to have a focal length of $f = 630$ μm assuming $\lambda = 850$ nm and free-space propagation. Both the microDOE and microlens arrays have an SiO_2 substrate thickness of 0.5 mm. The quality of the fabricated microDOE and microlens arrays will be assessed in a future investigation.

The optical system diagrammed in Fig. 5.20 is a simplified model of the actual 3-D PMCM neural network. Specifically, free-space optical propagation was assumed, when in reality multiple substrates with different indices of refraction are present in the 3-D PMCM. Following the method described in Appendix E, the modulator-microlens and microlens-detector spacings must be increased to account for the presence of the component substrates. Essentially, the appendix shows [Eq. (E.4), in which $\lambda' = \lambda$] that wavefront propagation through a distance Z in free space is equivalent to propagation through a distance Zn in a medium with index n. For example, it would take 2.2 mm of propagation through a $LiNO_3$ ($n \approx 2.2$) substrate for an optical wavefront to diffract equivalently to a 1.0 mm propagation through free space. (This analysis ignores constant phase factors.

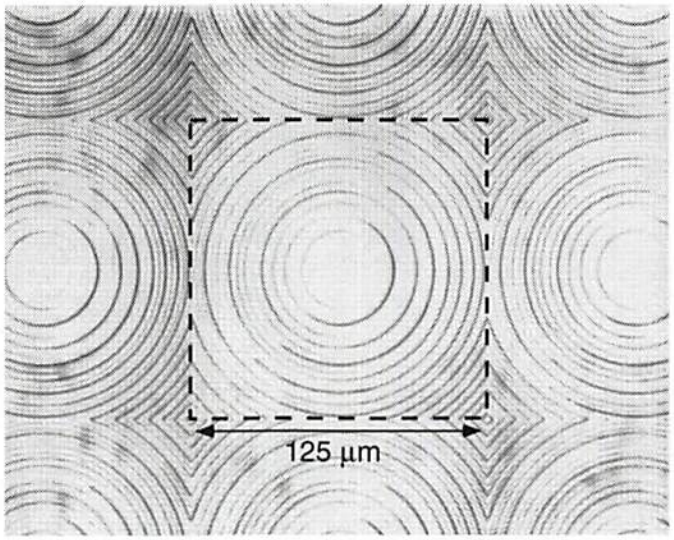


Figure 5.23: A photographed portion of a 10 × 10 microlens array (identification code: L9504Bf). Each microlens is 125 μm on a side.

The overall phase delay is not the same in the two cases.)

In Fig. 5.24 a 3-D PMCM neural network is shown, in which we assume a $LiNO_3$ optical power bus with a 1 mm substrate, an SiO_2 ($n \approx 1.5$) microlens array with a 0.5 mm substrate, and an SiO_2 microDOE array with a 0.5 mm substrate. The .3 mm and 1.1 mm spacings were chosen so that the equivalent free-space propagation remains at $Z_1 = 1103$ μm and $Z_2 = 1470$ μm. The photonic signal-shifting chip and programmable neural-network chip substrates were each assumed to be ≈ 1 mm thick. (Note that, as discussed earlier, only half of the modulators on the photonic signal-shifting chip are read out.) Under the above assumptions, the total (unpackaged) volume of the 3-D PMCM neural network is equal to 2 mm $\times$ 2 mm $\times$ 5.4 mm = 21.6 mm^3. This system volume is remarkably small when compared to typical free-space optical interconnection systems. For example, in [Yayla *et al.*, 1994] the authors reported a 3-D optically interconnected neural network with 64 weights (compared with 900 weights in our system described

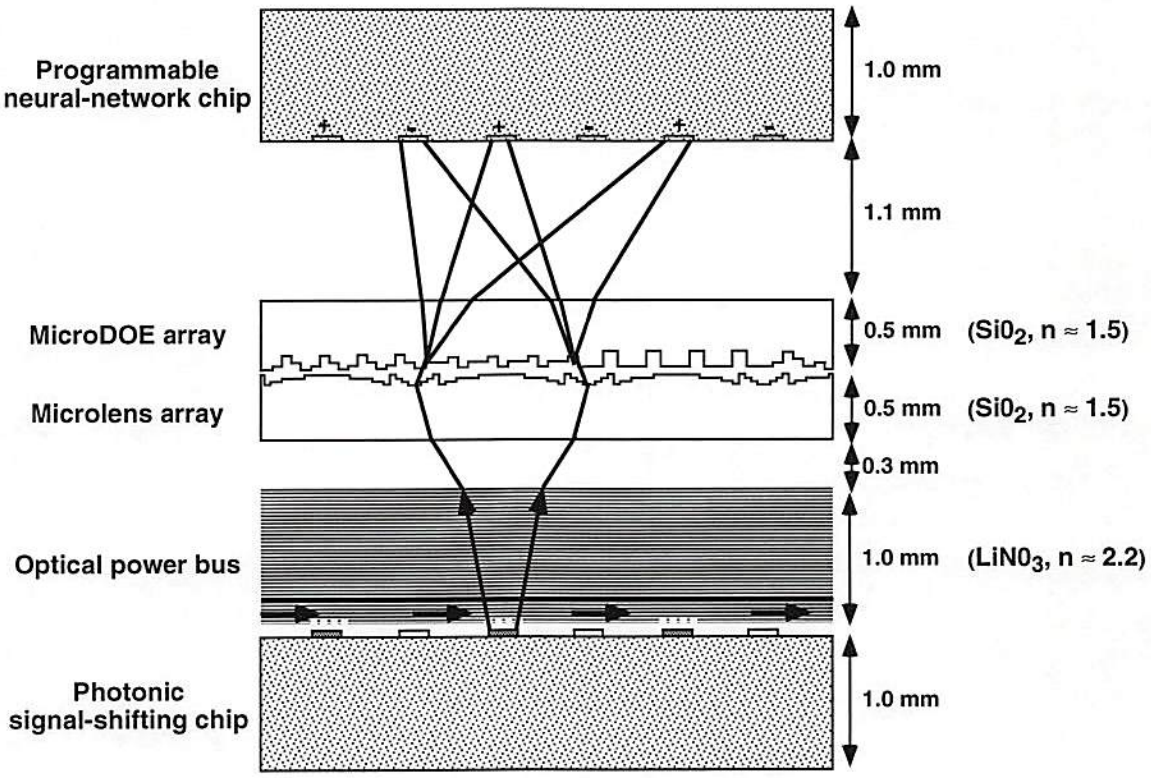


Figure 5.24: Optical propagation in a 3-D PMCM neural network. The photonic signal-shifting chip has an area of 2 $mm \times 2\ mm$, which implies that the total (unpackaged) 3-D PMCM neural-network volume is equal to 2 $mm \times 2\ mm \times 5.4\ mm = 21.6\ mm^3$.

above) that occupied a volume of $\approx 2370\ mm^3$.

5.7 Discussion

One of the lessons we learned during the design of the 3-D PMCM neural network is that the limited space-bandwidth product available for each microDOE is an important consideration in terms of how the system can be scaled up in future versions. If current trends continue, CMOS feature sizes will shrink dramatically in coming years, which will allow more transistors per unit area on the photonic signal-shifting chip and the programmable neural-network chip. This increase in

transistor density can be utilized by decreasing the size of each pixel and/or increasing the amount of circuitry in each pixel. If the pixels are decreased in size, then the microDOE's and microlenses will need to decrease in size proportionally. However, if the optical wavelength remains approximately constant, then microDOE design algorithms that take into account vector diffraction theory will need to be developed in order to take advantage of the technological improvements in photolithography. The simpler alternative is to keep the pixel size approximately constant, and to instead use the improvements in CMOS transistor density to put more electronic processing in each pixel on the photonic signal-shifting and programmable neural-network chips.

There are many interesting extensions of this work that could be pursued in the future. For example, development of Vertical Cavity Surface Emitting Laser (VCSEL) arrays is proceeding at a rapid pace both in academia and in industry. If VCSEL arrays could be successfully flip-chip bonded onto the CMOS signal-shifting circuitry (as are the GaAs optical modulators), then the optical power bus could be eliminated from the stack, and it would no longer be necessary to use an external laser source to optically power the PMCM. Another possibility for future research is to combine the microDOE and microlens arrays into one element. This approach requires more space-bandwidth product, but it eliminates one optical element from the PMCM stack. More practical issues such as the development of techniques for aligning, spacing, and attaching the PMCM components together also need to be addressed before a fully implemented system is realizable.

Chapter 6

A 3-D photonic multichip-module neural network: eye detection simulations

6.1 Introduction

An increasingly important application area for artificial neural network pattern recognition systems, both commercially and at the university research level, is the processing of human facial images [Samal and Iyengar, 1992] [Poggio and Beymer, 1996] [Johnson, 1996]. Depending on the specific application, this image processing may include the detection and location of one or more faces in a scene [Yang and Huang, 1994] [Sung and Poggio, 1994] [Rowley *et al.*, 1995] [Krueger *et al.*, 1996], the identification of an unknown input face (*e.g.*, [Lades *et al.*, 1993] [Pentland *et al.*, 1994]), facial expression recognition [Fellous, 1996], or the detection of specific facial features such as the eyes, nose, and mouth [Debenham and Garth, 1992] [Hegelin and Hewit, 1994]. The

main motivation for this increased research activity is the demand for more efficient human/computer interfaces, as well as more efficient biometric security systems that can help to authenticate automatic-teller-machine cards, credit cards, or the access codes to secure areas.

In this chapter, we will focus on the real-time detection and accurate location of the eyes in an image of a human face. To date, the demonstrated electronic hardware and software systems such as those described in [Debenham and Garth, 1992] and [Hegelin and Hewit, 1994] have been limited in speed, in detection accuracy, and to a relatively large system volume (*i.e.*, desktop computer systems). We present simulation results to show that a 3-D PMCM neural network, as described in the previous chapter, has the potential to accurately detect, in real time and with compact photonic hardware, the location of the two eyes in an image of a face. Such a real-time eye detection system would have several potential applications. In addition to serving as the first stage of a real-time face recognition or face detection system, a real-time eye detector could also be used, for example, either as a pre-processor for an authenticating retinal-scanning system, or as part of an eye-tracking human/computer interface to replace or augment input from a mouse-based pointer [Jacob, 1993] [San Jose Mercury News, 1996].

The eye detection task, described in the context of this chapter, is defined to be the decision whether or not an eye is located at the center of the input plane of an artificial neural network. The neural network has a 2-D input plane in which each neuron represents one pixel of the input image, and the network has a single output neuron that should yield a positive response when the 2-D input plane contains an example of a centered eye. The network can then be scrolled over each pixel location of the input image, outputting an eye-detection decision for

each of the pixel locations. Fig 6.1 illustrates an idealized example of eye detection using a scrolling neural network. As shown in the figure, the original gray-scale input image is transformed into an output image that is, ideally, completely black except for two bright pixels corresponding to the locations where the eyes have been detected.

This formulation of the eye detection task maps very well into the capabilities of 3-D PMCM neural network hardware, in that the required spatial and temporal bandwidths are well suited to the constraints imposed by current photonic technology. For example, most current multiple-quantum-well SLM's are limited to relatively small array sizes, such as the 10x20 photonic signal-shifting chip discussed in the previous chapter. However, these GaAs devices have a very impressive temporal bandwidth greater than 250 Mbits/sec per pixel [Goossen *et al.*, 1995]. A scrolling neural network system of the type described above illustrates one way to trade off some of the available temporal bandwidth for an effective increase in spatial bandwidth. While only a small fraction of the input image is analyzed at a given point in time, the network effectively analyzes the complete input image through the temporal multiplexing of its available spatial resources.

6.2 A space-variant edge detection neural model

For the simulations described in this chapter, a two-layer feed-forward neural network scrolls over the input image to perform the eye detection task. As shown in the 1-D representation of Fig. 6.2, the neural model is a fairly standard two-layer feedforward multi-layer perceptron [Rumelhart and McClelland, 1986]. The unique aspects of the model stem from the constraints imposed by 3-D photonic

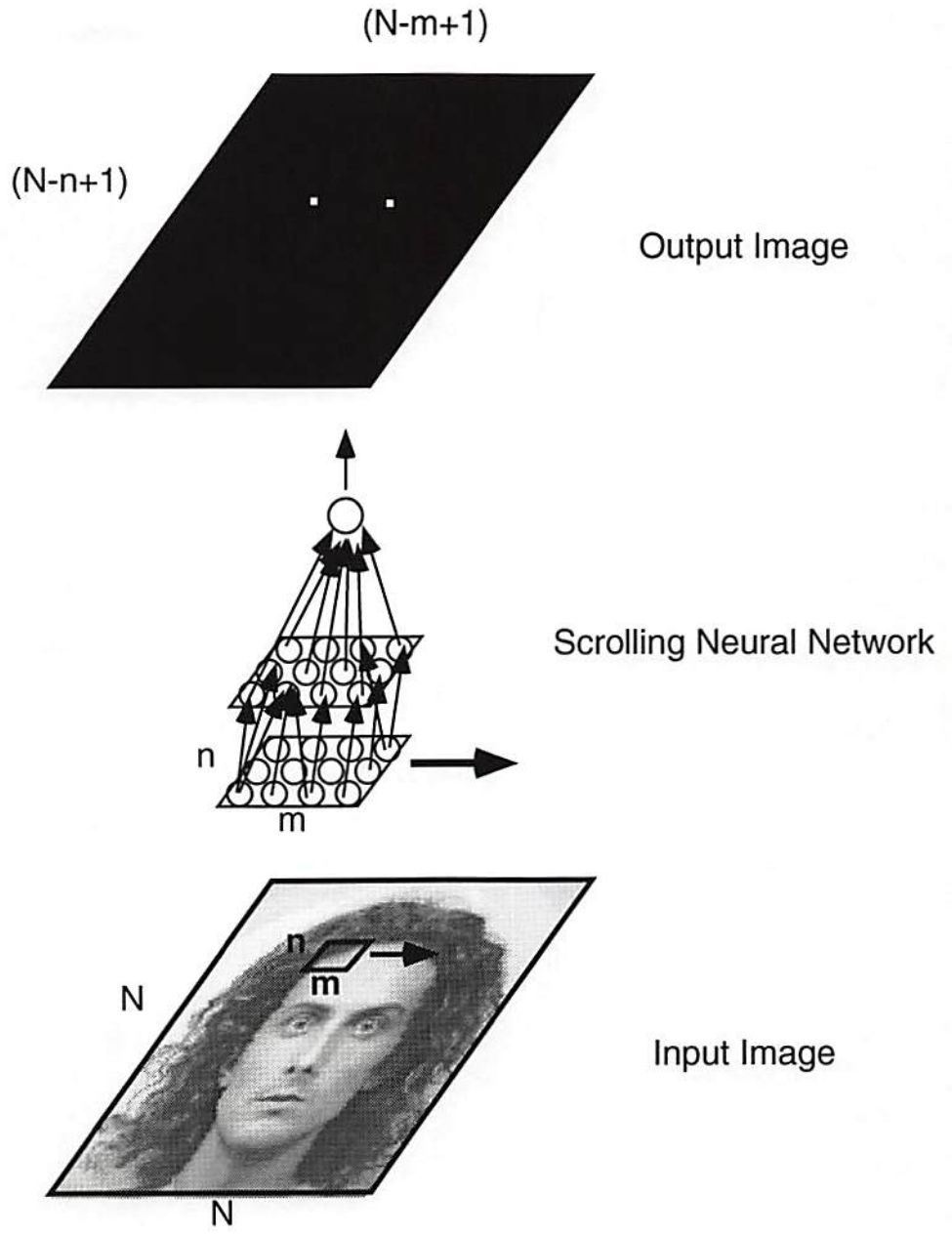


Figure 6.1: Eye detection using a scrolling neural network. The input image has NxN pixels; the neural network input window has nxm pixels. The neural network scrolls over all (N-n+1)x(N-m+1) possible window locations, outputting an eye-detection decision at each location.

multichip module hardware. Specifically, the first layer of weights is constrained to be fixed and locally connected, because the weights are designed to be implemented with etched micro-diffractive optical elements (microDOE's) that have a limited space-bandwidth product. The second layer of weights is allowed to be adaptive and globally connected, because this layer is designed to be implemented using silicon electronics. [The relatively small number of electronic weights (200) makes global electronic connections feasible.] Thus, the neural network as a whole can be trained to solve a variety of problems by training the second layer, but the first layer is fixed and should therefore serve as a useful feature-extraction layer for a whole class of problems rather than just a single problem. As a result, we designed the first layer of weights by hand rather than with a two-layer learning algorithm such as backpropagation.

The first layer of weights was designed to perform a space-variant edge detection (SVED) operation. Edge detection has been shown to be a useful pre-processing step for many neural network (*e.g.*, [Denker *et al.*, 1989]) and classical image processing (*e.g.*, [Pratt, 1991]) operations. A feed-forward neural network of the type shown in Fig. 6.2 can implement edge detection pre-processing by choosing the first layer of weights such that the hidden units respond to local pixel intensity differences in the input layer. The nonlinear activation function implemented in each hidden unit acts as the edge/non-edge decision function. In a conventional space-invariant edge detection computation, a single convolutional template is applied uniformly across the entire image (*i.e.*, the set of fan-in weights is the same for each hidden-layer neuron). In a *space-variant* edge detection computation, however, each hidden-layer neuron has a unique set of fan-in weights that determines the characteristics (*e.g.*, orientation) of the edge it is detecting.

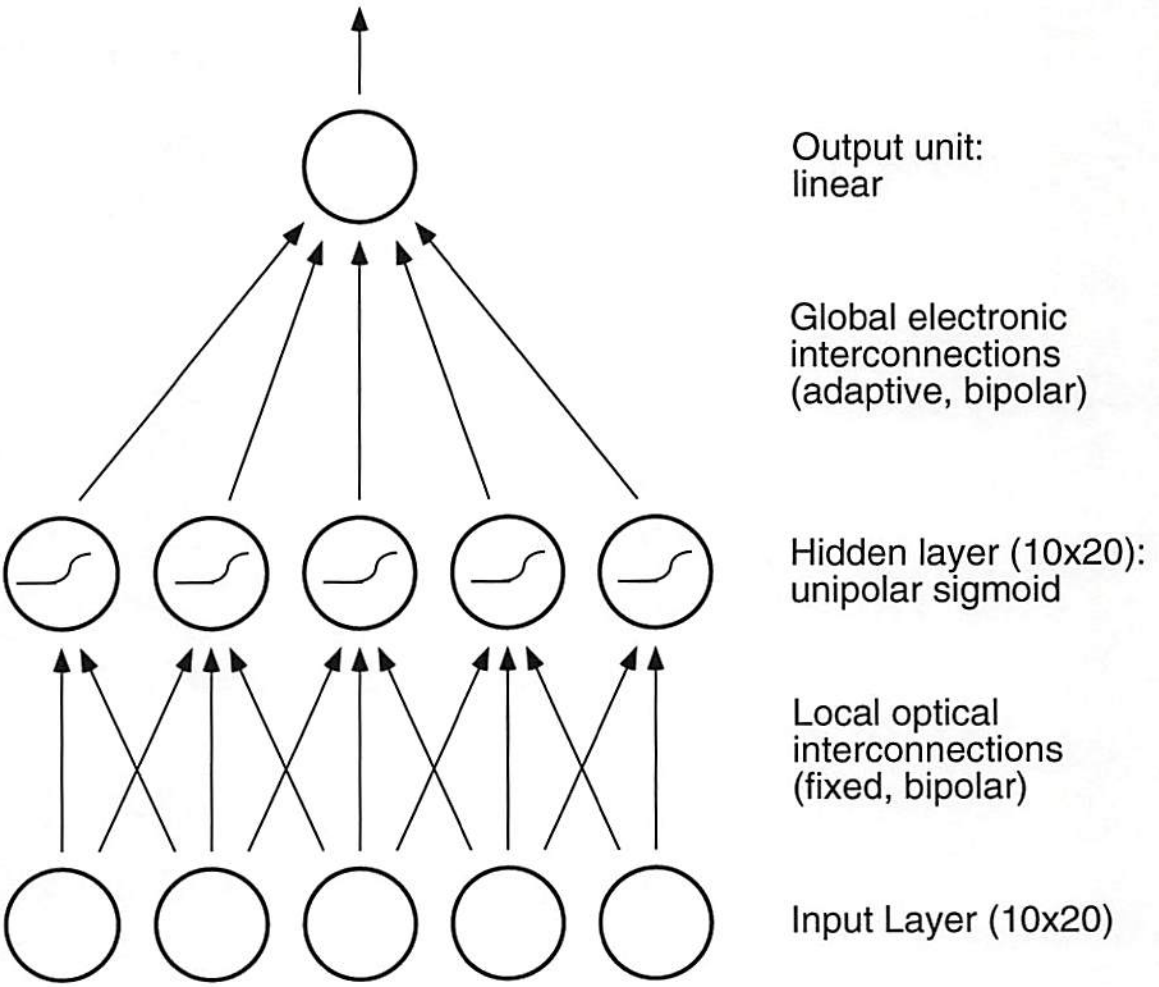


Figure 6.2: Neural network model used for eye detection. Input and hidden layers are 2-D planes (shown in 1-D for convenience).

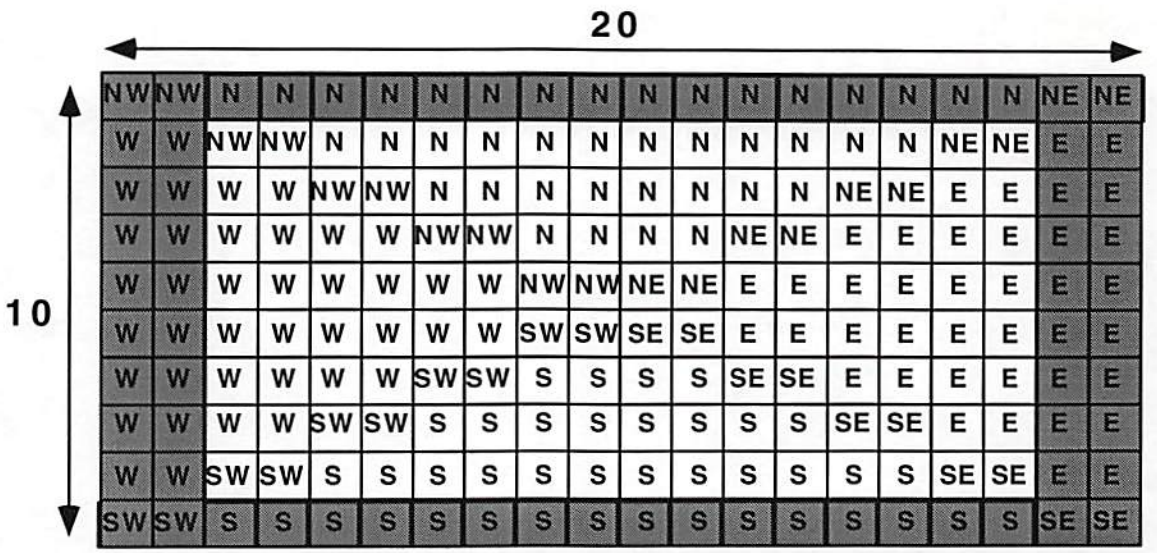
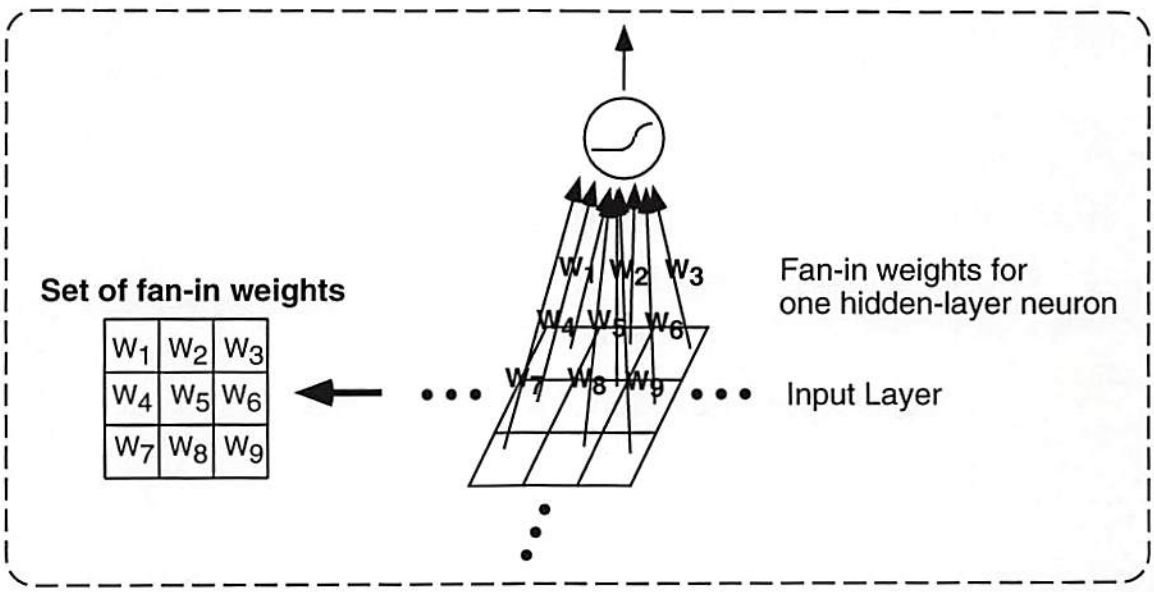


Figure 6.3: Space-variant edge detection hidden-layer map. Each square corresponds to one of the 10x20 hidden units, and each letter corresponds to a different set of edge-detection fan-in weights. These fan-in weights are defined in Figs. 6.4 and 6.5.

A complete hidden layer is then defined by assigning an appropriate set of fan-in weights to each hidden unit.

While space-variant processing requires many more computations than space-invariant processing when the network is scrolled over the entire input image (see Section 5.6.1 for more details), the additional flexibility provides the ability to give the neural network *a priori* information about the class of objects to be recognized. For example, in many applications it may be known that the objects of interest have edges that form simple convex shapes such as boxes or ellipses. Eye detection is an example of such an application, as are military target recognition tasks such as the detection of buildings and vehicles from overhead images of a scene [Rogers *et al.*, 1995]. By properly designing an SVED layer of weights, a network can be designed that filters out noise caused by edges that appear in unexpected orientations.



NW

0.8	0.8	0.8
0.8	-1	-1
0.8	-1	-1

N

0.5	0.5	0.5
0.5	0.5	0.5
-1	-1	-1

NE

0.8	0.8	0.8
-1	-1	0.8
-1	-1	0.8

W

0.5	0.5	-1
0.5	0.5	-1
0.5	0.5	-1

E

-1	0.5	0.5
-1	0.5	0.5
-1	0.5	0.5

SW

0.8	-1	-1
0.8	-1	-1
0.8	0.8	0.8

S

-1	-1	-1
0.5	0.5	0.5
0.5	0.5	0.5

SE

-1	-1	0.8
-1	-1	0.8
0.8	0.8	0.8

Figure 6.4: Space-variant edge detection fan-in weight definitions

Figure 6.3 shows the 10x20 space-variant design used for the eye-detection simulations. In the figure, a set of fan-in weights corresponding to an oriented edge detector is assigned to each hidden-unit. The letters in the figure correspond to the 8 compass directions: North West, North, South, etc. ("North" is taken as "up" in Figs. 6.4-6.5.) The edge detectors have been designed in the shape of "concentric rectangles". That is, edge detectors near the top of the input window respond only to edges oriented in the North direction, edge detectors near the right of the input window respond only to edges pointing in the East direction, etc. The same basic pattern repeats itself, from the outermost rectangle to the innermost rectangle. In this way, the system is given a certain amount of scale invariance together with the ability to filter out edges in unexpected regions of the input image (*e.g.*, a South edge near the left of the input window).

Figure 6.4 defines eight of the sixteen sets of edge-detection fan-in weights that were used to compose the first layer of weights. A hidden-layer neuron with fan-in weights from the "North" (N) mask, for example, will respond to its nearest-neighbor 3x3 region of the input image if the lowest row is darker than the mean of the upper two rows. (That is, the edge points in the northward direction, moving from dark to light.) The fan-in weights shown in gray along the borders of the figure correspond to special "border cases" in which a portion of the nearest-neighbor 3x3 window is outside of the 10x20 input window. Fig. 6.5 gives the definitions for these border cases. For these simulations, we restricted ourselves to 3x3 local fan-in in order to be consistent with the diffractive optical elements currently in fabrication, but 5x5 fan-in is also feasible and would significantly increase the variety of implementable edge detectors.

In order to minimize the effects of poor modulator contrast ratio, the sets of

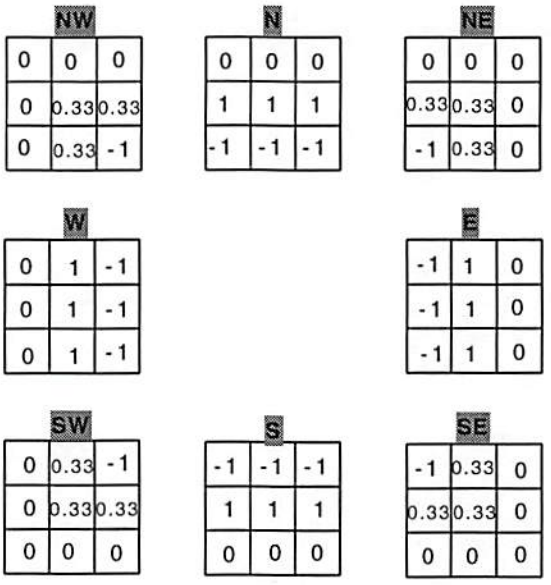


Figure 6.5: Space-variant edge detection border cases.

fan-in weights were designed such that the sum of all 9 weights in each set is equal to zero. Using this constraint the effect of a common bias signal is eliminated, as is shown using the relation

$$\rho_i = \sum_j w_{ij}(x_j + b_i) = \sum_j w_{ij}x_j + b_i \sum_j w_{ij}, \tag{6.1}$$

in which ρ_i is the input potential for hidden-layer neuron i, $\{w_{ij} : j = 1 \ldots 9\}$ is the set of fan-in weights to neuron i, and b_i represents a constant bias term for inputs fanning in to neuron i. Eq. (6.1) shows that the input potential ρ_i is unaffected by a bias signal b_i when the sum of the fan-in weights $\sum_j w_{ij}$ is equal to zero. This analysis does assume, however, that the modulator bias is constant over each 3x3 local fan-in region.

A unipolar shifted-sigmoid response at the hidden-layer was used as the transformation between neural input potential and output activation:

$$f(\rho) = 1/[1 + \exp(-(\rho/K - 5))], \tag{6.2}$$

in which $f(\rho)$ is the neural activation function, $K = 10$ is an empirically chosen normalization constant, and the bias constant 5 was chosen to satisfy $f(\rho = 0) \approx 0$. With this shifted sigmoid response, a hidden-layer neuron responds only when an edge is oriented in the correct direction. If the edge has the wrong polarity then ρ is negative, which yields an output $f(\rho)$ near zero. None of the hidden-layer neurons responds to a uniform input, because in this case $\rho = 0$ [using Eq. (6.1)], and $f(\rho = 0) \approx 0$ by design. The choice to use a unipolar sigmoid in these simulations, rather than a bipolar sigmoid, was made for two reasons. First, a unipolar sigmoid is easier to implement in the electronic analog hardware. Second, it is more convenient to visually inspect the output of the hidden layer when it is unipolar (see Sec. 6.4 for examples).

6.3 Neural network training

The first layer of the neural model, as described above, is a fixed feature extraction layer that can be implemented using optical hardware. This layer is, by design, not specific to a given problem. In order to solve a particular problem, the second (adaptive) layer of weights in Fig. 6.2 needs to be specified. In the following, we describe how the second layer of weights was trained using a database of facial images.

6.3.1 Face database

A database of faces was obtained from Dr. Jean-Marc Fellous, who at the time was part of Dr. Christoph von der Malsburg's research group at USC. Dr. Fellous obtained the database by frame grabbing a 256x256 image of each subject's face using a CCD video camera. The database is shown in Fig. 6.6. Each of the 26 faces in the database has 128x128 pixels with 8 bits of gray-scale resolution. The original 256x256 images obtained from Dr. Fellous were reduced to 128x128 pixels, so that the eyes in each image were able to fit into a 10x20 pixel window in accordance with the photonic signal-shifting chip hardware described in the previous chapter. This image scaling was accomplished by reducing each 2x2 square block of the original 256x256 image down to a single pixel with an intensity equal to the average of the 2x2 pixel intensities. The resultant degradation in the apparent quality of the images was minimal.

Although an attempt was made to obtain the images under controlled illumination and optical imaging conditions, the images were obtained on different days and times. Thus, significant illumination differences between the faces in the database are evident. There are also scale differences between some of the faces; for example, the face of the woman shown in the right-most column and third row down is noticeably larger than that of the man shown directly beneath her. Non-idealities such as those shown in the database are unavoidable in most real-world pattern recognition scenarios. This database is thus a fairly realistic test of the robustness of our neural model in the presence of real-world distortions and noise.

Figure 6.6: Database of faces. Each face is an 8-bit gray-scale, 128x128 image.

6.3.2 Extraction of training data

A total of 52 examples of eyes (26 faces $\times$ 2 eyes/face) were extracted from the face database, as shown in Fig. 6.7(a). We extracted these images by visually inspecting the images to determine our best estimate for the center of each eye. These 52 eyes were then transformed into the space-variant edge detected (SVED) images shown in Fig. 6.7(b), using the SVED interconnection weights and hidden-unit sigmoids described previously in Sec. 6.2. Similarly, a total of 10,400 examples of "non-eyes" (26 faces $\times$ 400 non-eyes/face) were extracted from the face database, where a non-eye is defined to be a randomly selected 10x20 region of the input image that is *not* centered in either of the 5x5 windows surrounding the two eyes. That is, a non-eye can include a portion of an off-centered eye, but it must be shifted off center by at least 3 pixels in either the horizontal or vertical direction. An example set of 400 non-eyes extracted from one face is shown in Fig. 6.8, along with the corresponding set of 400 SVED versions of these non-eyes.

Comparing the SVED eye images in Fig 6.7 with the SVED non-eye images in Fig. 6.8, the most striking difference between the two is that there are many more black pixels in the non-eye images. One reason for this difference is that the non-eye locations were chosen randomly, and as a result there are many "boring" images that have a nearly uniform intensity. Furthermore, even when the non-eyes do have edges, they are randomly oriented with respect to the SVED hidden layer and therefore only a fraction of them are detected. The eyes, on the other hand, have many more edges that are oriented such that the SVED hidden-layer detects them. The fact that the SVED eyes and non-eyes appear to be statistically separable is quite promising in terms of training the second layer of weights in Fig. 6.2.

A more detailed viewing of the raw-data and SVED non-eye images also verifies that the space-variant edge detection layer is indeed sensitive to edge orientation. For example, in Fig. 6.8 the bottom-left image has an edge in the raw data; however, no edge appears in the SVED image because it is a northeast edge located in the southeast portion of the image. Conversely, an edge does appear in the leftmost SVED image third from the bottom, because in this case the edge is oriented in the correct direction given its spatial position. For the SVED eye images, it is a bit more difficult to see the edge detection because of the large number of edges in such small images. Also, light reflections from the pupil can cause spurious edges to appear in the center of the eye (*e.g.*, the top pair of SVED eye images).

6.3.3 Weighted least squares training

Each of the eye and non-eye images discussed above were transformed into column vectors by raster scanning the 10x20 images into vectors of length 200. Four sets of training vectors were obtained: a set of raw-data eye vectors $\{\mathbf{e}_i : i = 1 \ldots 52\}$, a corresponding set of SVED eye vectors $\{\mathbf{e}'_i : i = 1 \ldots 52\}$, a set of raw-data non-eye vectors $\{\mathbf{n}_i : i = 1 \ldots 10,400\}$, and a corresponding set of SVED non-eye vectors $\{\mathbf{n}'_i : i = 1 \ldots 10,400\}$. All vectors were augmented with an additional component with value 1 for biasing [Duda and Hart, 1973], bringing the total length to 201 components. Only the SVED eye and non-eye vectors were used for the two-layer neural network training discussed in the remainder of this section; the raw-data eye and non-eye vectors are discussed later in Sec. 6.4.2.

In order to fully utilize a limited amount of training data, the cross-validation

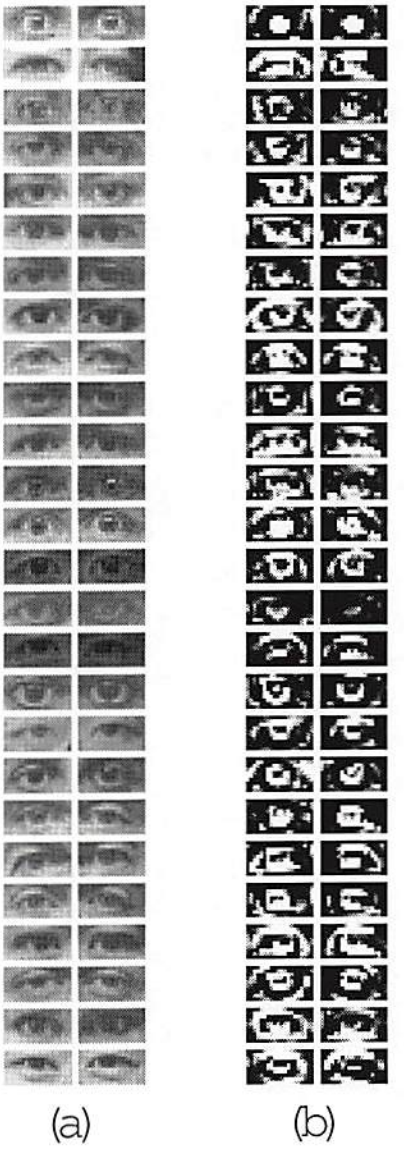
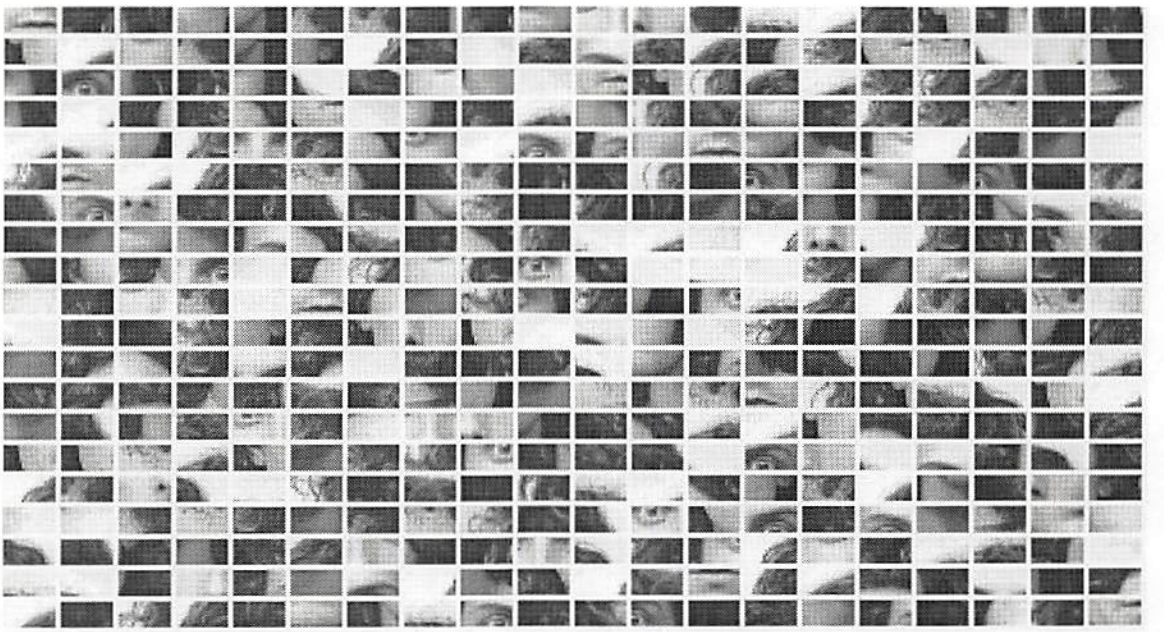
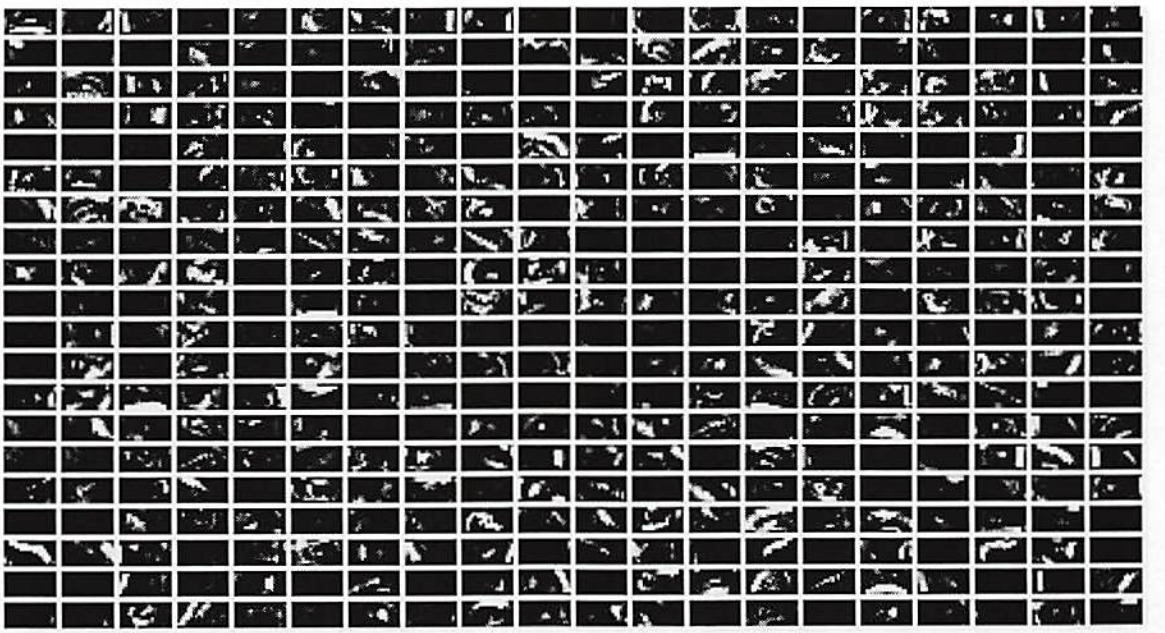


Figure 6.7: Extraction of eyes from database: (a) raw data, (b) space-variant edge detected. Each eye (both raw and edge detected) is a 10x20 image.



(a)



(b)

Figure 6.8: Extraction of non-eyes from database; (a) raw data, (b) space-variant edge detected. Each non-eye (both raw and edge detected) is a 10x20 image. Four hundred non-eyes were randomly chosen from each image.

procedure known as "leave-one-out" [Fukunaga, 1990] was employed during training. This procedure eliminates the need to divide the available data into independent training and testing sets. Instead, a different neural network is trained for each image in the database, where the data from the image being tested is not used during training (*i.e.*, one image is "left out" during each training session). In this way, all 26 images in the database can be used for testing, and a different set of 25 training images is used for each training session.

For each training session, the 50 SVED eye vectors (25 training faces $\times$ 2 eyes/face) and 10,000 SVED non-eye vectors (25 training faces $\times$ 400 non-eyes/face) were stacked into a matrix $\mathbf{M}'$ given by

$$\mathbf{M}' = \left[\begin{array}{c} \mathbf{e}'^{T}_{1} \\ \mathbf{e}'^{T}_{2} \\ \vdots \\ \mathbf{e}'^{T}_{50} \\ \hline \mathbf{n}'^{T}_{1} \\ \mathbf{n}'^{T}_{2} \\ \vdots \\ \mathbf{n}'^{T}_{10,000} \end{array}\right], \tag{6.3}$$

in which T denotes the matrix transpose operation. The second layer of weights was then determined using a weighted-least-squares solution for $\mathbf{w}$ in the expression

$$\mathbf{M}'\mathbf{w} = \mathbf{t}, \tag{6.4}$$

where $\mathbf{w}$ is a raster-scanned representation of the second layer of weights and $\mathbf{t}$

is a vector of target values (+1 for the eyes, -1 for the non-eyes). The weighted-least-squares solution of Eq. (6.4) can be expressed as [Mendel, 1987]

$$\hat{\mathbf{w}} = (\mathbf{M}'^T\mathbf{D}\mathbf{M}')^{-1}\mathbf{M}'^T\mathbf{D}\mathbf{t}, \tag{6.5}$$

where $\hat{\mathbf{w}}$ is the weighted-least-squares solution for $\mathbf{w}$, and $\mathbf{D}$ is an arbitrary diagonal weighting matrix used to define the relative importance of each of the components in the target vector $\mathbf{t}$. For these eye detection simulations, the weighting matrix $\mathbf{D}$ was employed to compensate for the vast difference between the number of available eye vectors (50) and the number of available non-eye vectors (10,000). As explained in Appendix A, we compensated for this difference by setting the diagonal elements of $\mathbf{D}$ corresponding to the eye vectors (the first 50 rows) equal to the ratio $10,000/50 = 200$ and set the remaining diagonal elements (the final 10,000 rows) equal to 1. This choice for the weighting matrix $\mathbf{D}$ leads to the same solution $\hat{\mathbf{w}}$ that is obtained with the (unweighted) least-squares solution of Eq. (6.4), provided that each of the eye vectors is replicated 200 times in forming the matrix $\mathbf{M}'$ and target vector $\mathbf{t}$. Thus, in a sense we equalized the number of eyes to the number of non-eyes using the $\mathbf{D}$ matrix.

6.4 Simulation results

6.4.1 Space-variant edge detection results

Having determined both layers of interconnection weights in the neural network model of Fig. 6.2, the performance of each of the leave-one-out-trained networks

was assessed by scrolling the network (as diagrammed in Fig. 6.1) over the complete facial image that was left out during training. These results are shown in Fig. 6.9, where the output images are arrayed in the same fashion as shown in Fig. 6.6. When the neural network output is less than or equal to zero (indicating a non-eye) a black pixel is drawn. For outputs greater than or equal to zero, a gray-scale pixel is drawn with an intensity proportional to the output value. (Each image was normalized so that the highest output value in the image corresponds to a maximally bright pixel.) Defining a "false positive" as a pixel that has an intensity greater than 0 and is outside of the 5x5 regions centered at the two eyes, the SVED neural network yielded a very low false-positive rate of 0.73% [false-positive rate = (# false positives)/(# pixels in output image outside of the 5x5 eye regions)] . At the same time, there was only 1 "false negative" out of 52 eyes ($\approx 2\%$), where a false negative is defined as a case in which all 5x5 pixels are black in the region surrounding an eye. Note that the 5x5 regions used in these performance definitions correspond to the definition for a non-eye that we used during training.

In order to find the two most likely positions of the eyes in each image, a digital post-processing algorithm was employed. The algorithm uses *a priori* information specific to the eye detection problem. In particular, the estimated locations of the two eyes are constrained to be within a certain horizontal and vertical distance from each other. These horizontal and vertical distance constraints were determined by inspection of the training data. In the horizontal direction, it was found that the distance between the two eyes could be loosely bounded by the range $|\Delta x|_{min} = 20$ and $|\Delta x|_{max} = 40$. In the vertical direction, the distance was bounded from above by $|\Delta y|_{max} = 3$. These bounds describe the two rectangular

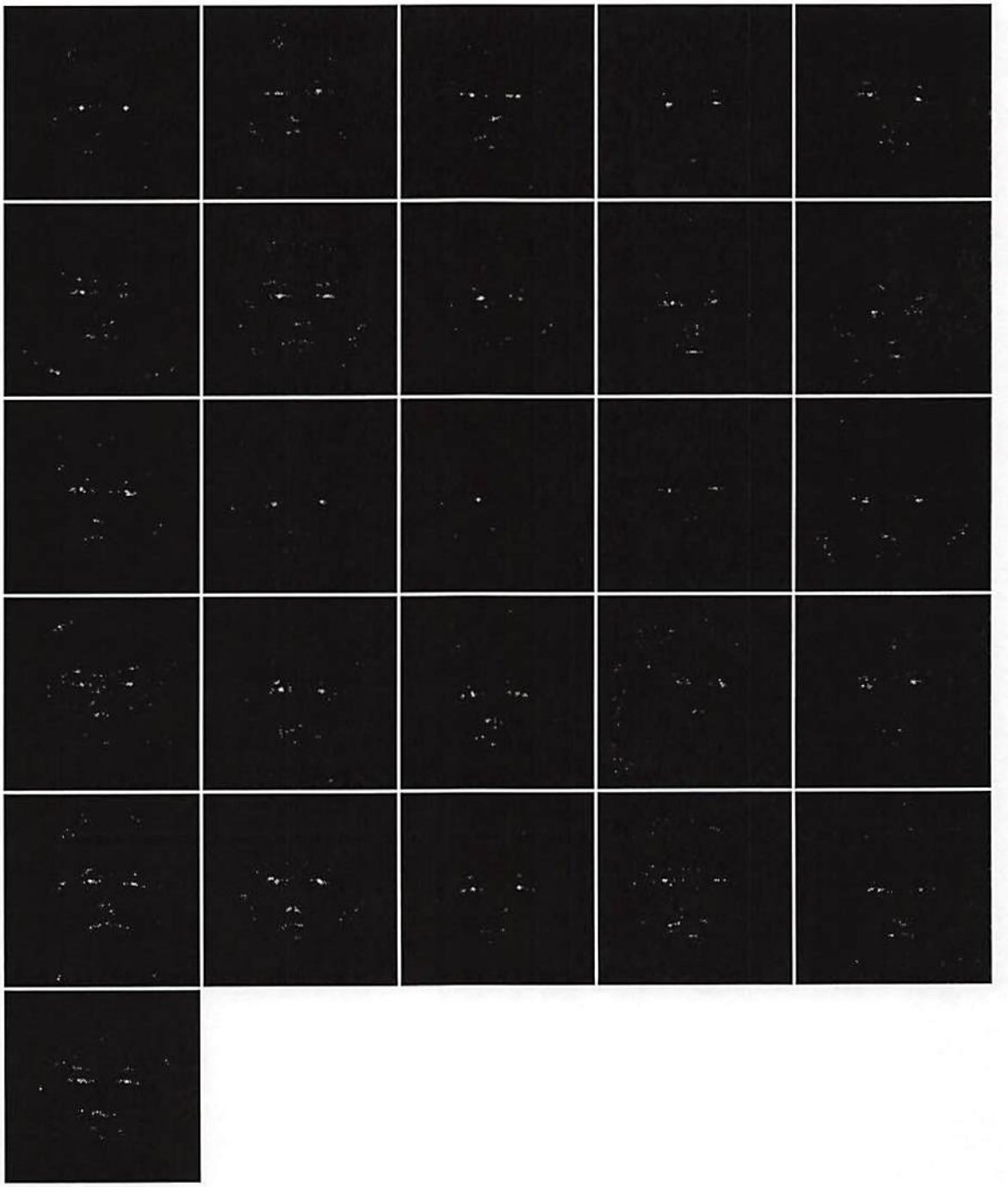


Figure 6.9: SVED neural network output (no post-processing).

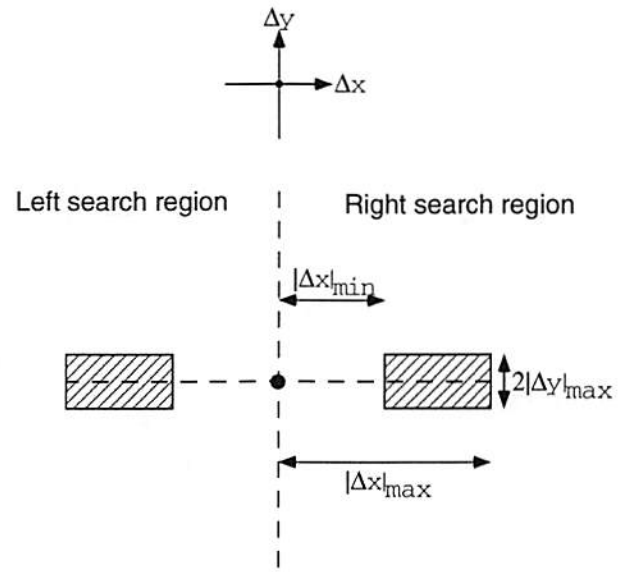


Figure 6.10: Two rectangular search regions for the second eye. The origin represents a candidate eye location; the two shaded regions are searched for the best candidate matching eye location.

regions shown in Fig. 6.10. The relatively large variation in the horizontal direction was necessary because of the significant differences in scale between some of the images, while the vertical bound could be much tighter because head rotation was not as significant an issue.

The post-processing algorithm is described by the following steps. First, the 10 brightest pixels in a given output image serve as a set of candidate eye locations. For each of these 10 candidate eye locations, the brightest pixel among those in either of the two rectangular regions defined above serves as the most likely location for the second eye. A set of 10 scores is thus obtained, where the score for a pair of eye locations is the average of the two candidate pixel intensities. The pair of candidate eye locations with the highest score is the final output of the post-processing algorithm.

Fig. 6.11 shows the results of applying the post-processing algorithm to the

output images given in Fig. 6.9. In the figure, an estimated eye location is drawn as a 5x5 block of pixels, corresponding to the goal of locating the center of an eye within an accuracy of a 5x5 window. If the estimated eye location is indeed within the stated 5x5 window, then the block of pixels is drawn in white; otherwise, the block of pixels is drawn in black. The overall performance of the two-layer neural network and post-processing algorithm is quite impressive: out of a total of 52 eyes tested, there are only 2 errors, yielding a post-processed error rate of $\approx$ 4%. It should be noted that after post-processing, there is no longer a distinction between false-positive and false-negative errors as there was in the previous discussion; post-processing of an image always results in two estimated eye locations, of which there are either 0, 1, or 2 errors.

The total computation time was approximately 20 seconds per image on a SPARC 10 workstation. (This is feed-forward processing time, not the time required for training the neural network.) Out of this 20 seconds, greater than 19 seconds was occupied by the multiplications, additions, and sigmoid calculations required by the feed-forward neural network processing. Less than 1 second was occupied by the digital post-processing algorithm. (Exact times were not measured.) Thus, the computationally intensive processing is well suited to parallel optical computations that can be performed by 3-D PMCM neural-network hardware, while the post-processing algorithm can be readily handled by conventional electronic hardware.

Comparing these detection results to those reported in the literature, the recent work from Hagelin & Hewitt at the University of Dundee (UK) [Hegelin and Hewit, 1994] is the most relevant. Their goal was also to find the eyes in an image of a face using a neural network that scrolls over the input image.

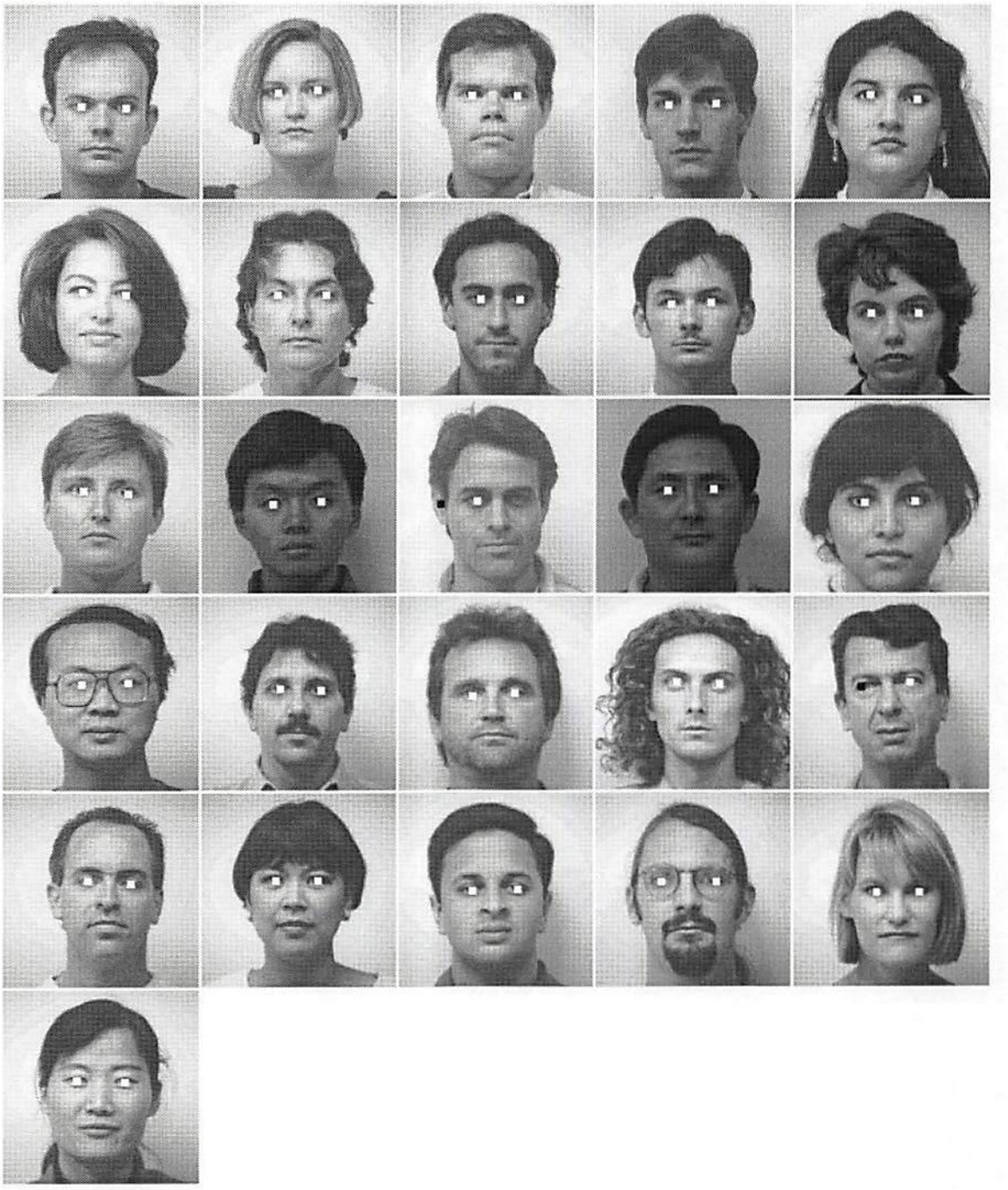


Figure 6.11: Post-processed SVED neural network output. A white square indicates a correct eye location (within a 5x5 window of actual center), a black square indicates an incorrect eye location. Post-processed error rate: $2/52 \approx 4\%$.

They trained a radial basis function neural network using an extension of the restricted coulomb energy (RCE) [Reilly *et al.*, 1982] [Debenham and Garth, 1992] learning algorithm. In their first experiments, the facial images of real subjects were taken under varying ambient lighting conditions and camera aperture settings (as were the images used in our database). The authors reported that the results were not satisfactory, although no quantitative measure of performance was given. To overcome this problem, they then used digitized student photographs (presumably taken by a professional photographer) as their training and testing database. Under these more controlled conditions, the authors reported an error rate of 2.5% (with the same 5x5 region definition of "error" that we used) using 20 training images and 40 testing images. The processing time was 10 seconds per image on a 486DX personal computer, although the authors claim that this time could be reduced to 0.12 seconds with a TMS320C30 DSP chip.

6.4.2 Single-layer network results

It is also useful to compare the eye detection performance of the SVED two-layer neural network model to that of a simpler single-layer network. This comparison is an important one to make, because a single-layer network is significantly easier to implement in real-time using standard digital electronic hardware (*e.g.*, a DSP chip). Fig. 6.12 diagrams a single-layer network model. It is a fully linear model with 10x20 raw-data inputs and one output unit. A performance comparison between the two models helps to quantify the relative importance of the computationally demanding space-variant edge detection layer present in the two-layer neural network model.

The single layer of weights was trained in a fashion very similar to that used

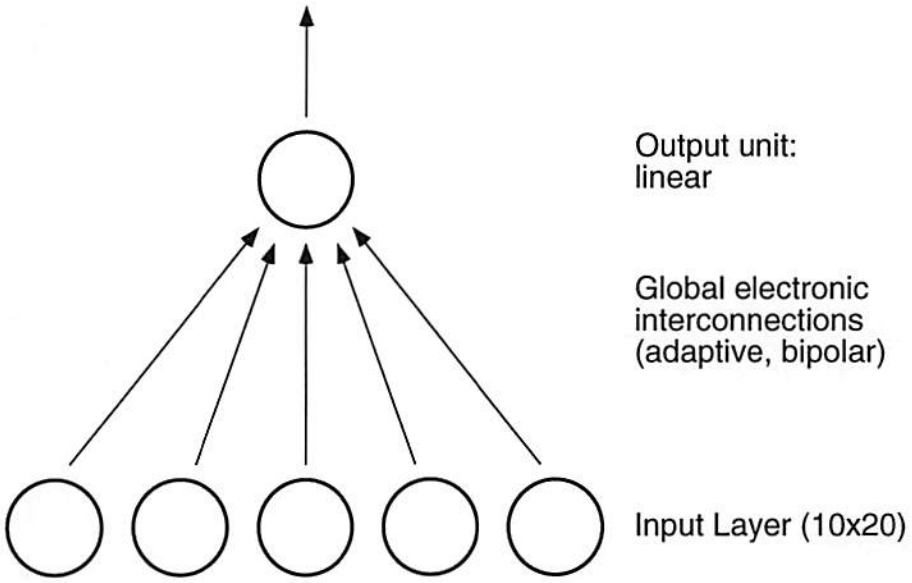


Figure 6.12: Single-layer network model. Input to network is raw gray-scale image data.

for the second layer of the two-layer network. Referring to Sec. 6.3.3, the only difference was that rather than using the SVED matrix $\mathbf{M}'$, which was formed by stacking the SVED eye and non-eye vectors $\{\mathbf{e}'\}$ and $\{\mathbf{n}'\}$, the raw data matrix $\mathbf{M}$ (formed by stacking the $\{\mathbf{e}\}$ and $\{\mathbf{n}\}$ vectors) was used to determine the unknown weight vector $\mathbf{w}$ in Eq. (6.5). The diagonal weighting matrix $\mathbf{D}$ and target vector $\mathbf{t}$ were the same for both models. Once again, the leave-one-out cross-validation method was used in order to avoid dividing the available data into training and test sets, yielding a total of 26 single-layer weight vector solutions.

Using the resultant set of single-layer weight vector solutions, each network was scrolled over the image that was left out during training. The output images are shown in Fig. 6.13. As shown in the figure, the single-layer network did solve the eye detection problem to a certain extent, but the performance was not as good as it was for the SVED neural network case. In particular, the performance on the faces with glasses was poor, and the single-layer network was not as able

to suppress spurious detections caused by hair line edges.

In order to adequately compare the single-layer and SVED neural network outputs (before post-processing) shown in Figs. 6.13 and 6.9, a quantitative measure is needed to determine how well the neural networks have separated the eye locations from the many non-eye locations. A reasonable quantitative measure would be a comparison between the false-positive and false-negative error rates associated with each of the networks. Using this performance measure, the single-layer network yielded approximately twice the number of false positives (1.4%) as did the SVED neural network (0.72%), while both models yielded the same number of false negatives (1 eye). Although this measure does correctly indicate that the SVED neural network had superior performance, the difference between the two models is underestimated because the relative intensities of the false-positive pixels (compared to the intensities in the 5x5 regions surrounding the correct eye locations) have been ignored in the comparison.

A better quantitative measure of the separation between eyes and non-eyes is described by the following: find the maximum pixel intensity for each of the two 5x5 eye regions, use the lesser of the two maxima as a threshold, and then count the number of "significant false-positive" pixels that have an intensity greater than the threshold. In this way, the comparatively weak false positives, which can be filtered out by a reasonable post-processing algorithm, are ignored in the performance measure. Using this revised performance measure, the total number of significant false positives, averaged over all 26 images, was approximately 1.7 pixels for the SVED neural network compared to 10.0 pixels for the single-layer network. This factor of ≈ 6 differentiating the two networks is supported by the post-processed results shown in Fig. 6.14. As shown in the figure, when

Figure 6.13: Single-layer neural network output (no post-processing).

the post-processing algorithm described previously is applied to the single-layer output images, the post-processed error rate is $\approx 27\%$ compared to $\approx 4\%$ for the SVED neural network output images. In terms of computation time on a SPARC 10 workstation, as expected the single-layer network required much less time to detect the eyes: including post-processing, the single layer network took less than 1 second, compared to approximately 20 seconds for the SVED neural network.

6.5 Binary image eye detection

The simulation results presented above have assumed that the scrolling neural network used for eye detection has a 2-D analog input plane. That is, the CCD-version of the photonic signal-shifting chip described in the previous chapter has been assumed. A valid question is then, how much will the performance degrade if the binary signal-shifting chip described in the previous chapter is used instead? Clearly one would expect there to be some degradation, given that the amount of image information present at the input plane has been drastically reduced. In the remainder of this section, we will quantify the performance degradation by repeating the SVED and single-layer neural network eye detection simulations for the case of binary faces rather than gray-scale faces.

The original gray-scale face database was transformed into a binary face database by setting the pixel intensity threshold equal to an empirically determined fraction of the mean pixel intensity; it was empirically determined that a reasonable fraction is 68%. That is, if the intensity of a pixel is greater than or equal to 68% of the average pixel intensity over the entire face, then the pixel is set to "white"; otherwise, the pixel is set to "black". The resultant binary images

Figure 6.14: Post-processed single-layer network output. A white square indicates a correct eye location (within a 5x5 window of actual center), a black square indicates an incorrect eye location. Post-processed error rate: $14/52 \approx 27\%$

are shown in Fig. 6.15. Due to the significant illumination differences between the original gray-scale images, as well as differences in skin complexion, hair color, and amount of hair, some of the binarized images are inadequate representations of the original gray-scale images. For instance, the eyes are not clearly visible in the image found in the fourth row down and second column from the right. It is possible that a more sophisticated thresholding technique (for example, basing the threshold on the gray-scale pixel intensity histogram of the original image [Debenham and Garth, 1992]) will yield better performance.

Following the same procedure described previously, 2 eyes and 400 non-eyes were extracted from each image in the binary database. These eyes and non-eyes were then transformed into the SVED eyes and non-eyes that were used for training the second layer of weights. Figs. 6.16 and 6.17 show the binary training data, before and after space-variant edge detection. For the binary case, we increased the empirically chosen normalization constant in Eq. (6.2) from $K = 10$ to $K = 50$, in order to compensate for the increased brightness of the binary images.

Training the second layer with the extracted SVED eyes and SVED non-eyes, and then scrolling over each of the binary faces using the leave-one-out technique, yielded the direct neural-network output results shown in Fig. 6.18. Post-processing these direct output images resulted in a post-processed error rate of 46%. (The post-processed images are not shown because the black and white squares that indicate the estimated eye locations are not easily visible when overlayed on binary images.) Clearly, there is a noticeable degradation in performance when the binary images are used as input, as the post-processed error rate increases by an order of magnitude. The single-layer network performance using

Table 6.1: **Summary of eye-detection results**

Performance measure	Gray-scale		Binary	
	SVED	Single-layer	SVED	Single-layer
False-positive rate	0.72%	1.4%	1.8%	1.6%
False negatives (# eyes)	1	1	1	2
Sig. false positives (# pixels/image)	1.7	10.0	24.8	22.9
Post-processed error rate	4%	27%	46%	46%
Total processing time (sec.)	≈ 20	< 1	≈ 20	< 1

binary input images was the same as the above SVED results: the post-processed error rate is 46%

6.6 Summary of results and discussion

Table 6.1 summarizes the results presented in this chapter. The basic conclusion to be drawn is that space-variant edge detection is a time consuming pre-processing operation when implemented on a digital computer, but it yields impressive detection results for gray-scale images. The number of significant false positives and the post-processed error rate decreased by a factor of 6 in comparison to the single-layer network case. The use of 3-D PMCM neural-network hardware potentially allows this computationally intensive algorithm to be implemented in real time, in an optoelectronic module with a volume on the order of tens of cubic millimeters (see previous chapter for details).

The table also shows, however, that the binary-image results were significantly inferior to the gray-scale results. One conclusion to be drawn is that it makes sense to pursue development of the CCD-version of the photonic signal-shifting chip discussed in the previous chapter. Another approach is to use the

Figure 6.15: Database of binary faces. Each face is a 128x128 binary image.

Figure 6.16: Extraction of eyes from binary database: (a) raw binary data, (b) space-variant edge detected. Each eye (both raw and edge detected) is a 10x20 image.

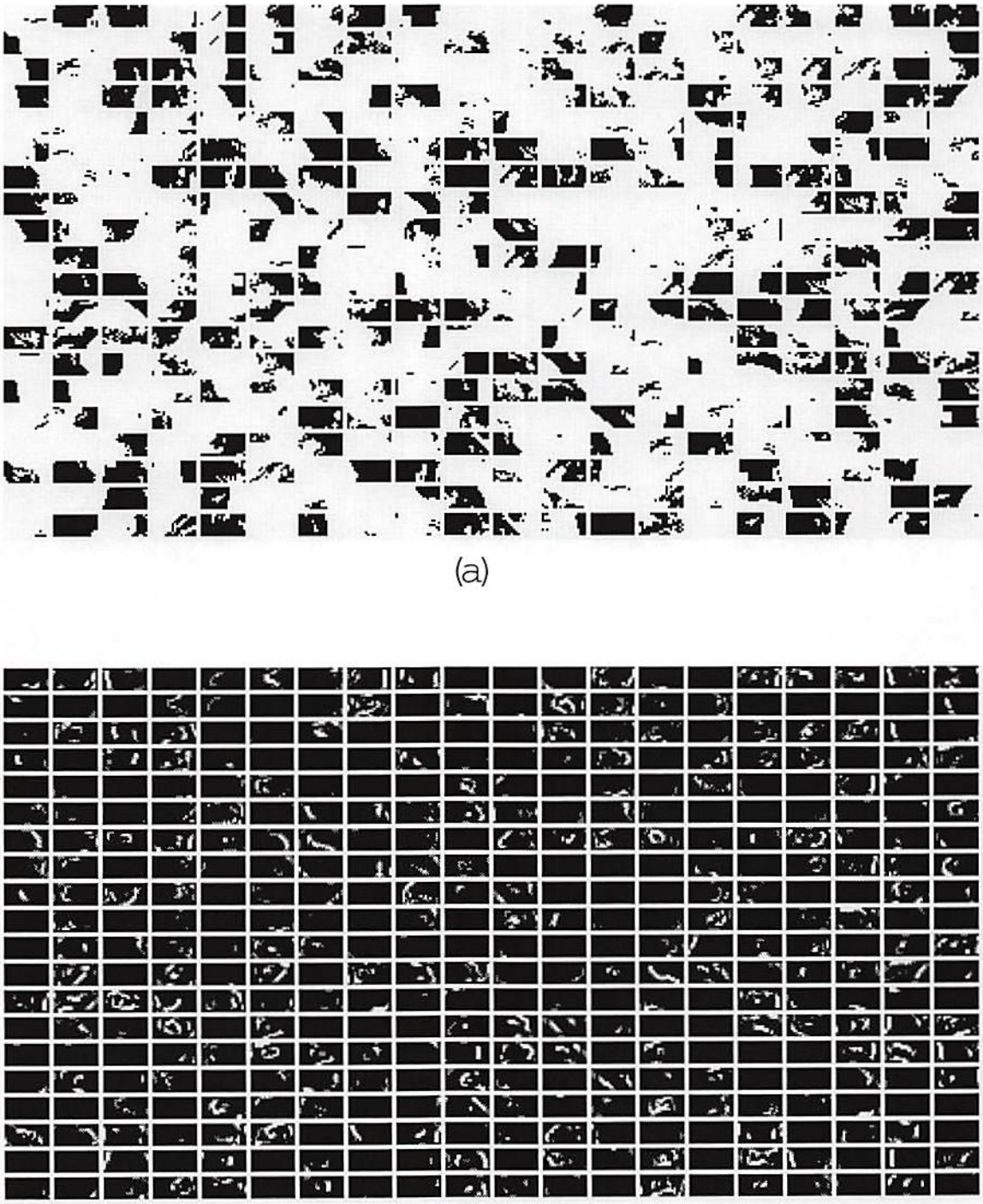


Figure 6.17: Extraction of non-eyes from binary database; (a) raw binary data, (b) space-variant edge detected. Each non-eye (both raw and edge detected) is a 10x20 image. Four hundred non-eyes were randomly chosen from each image.

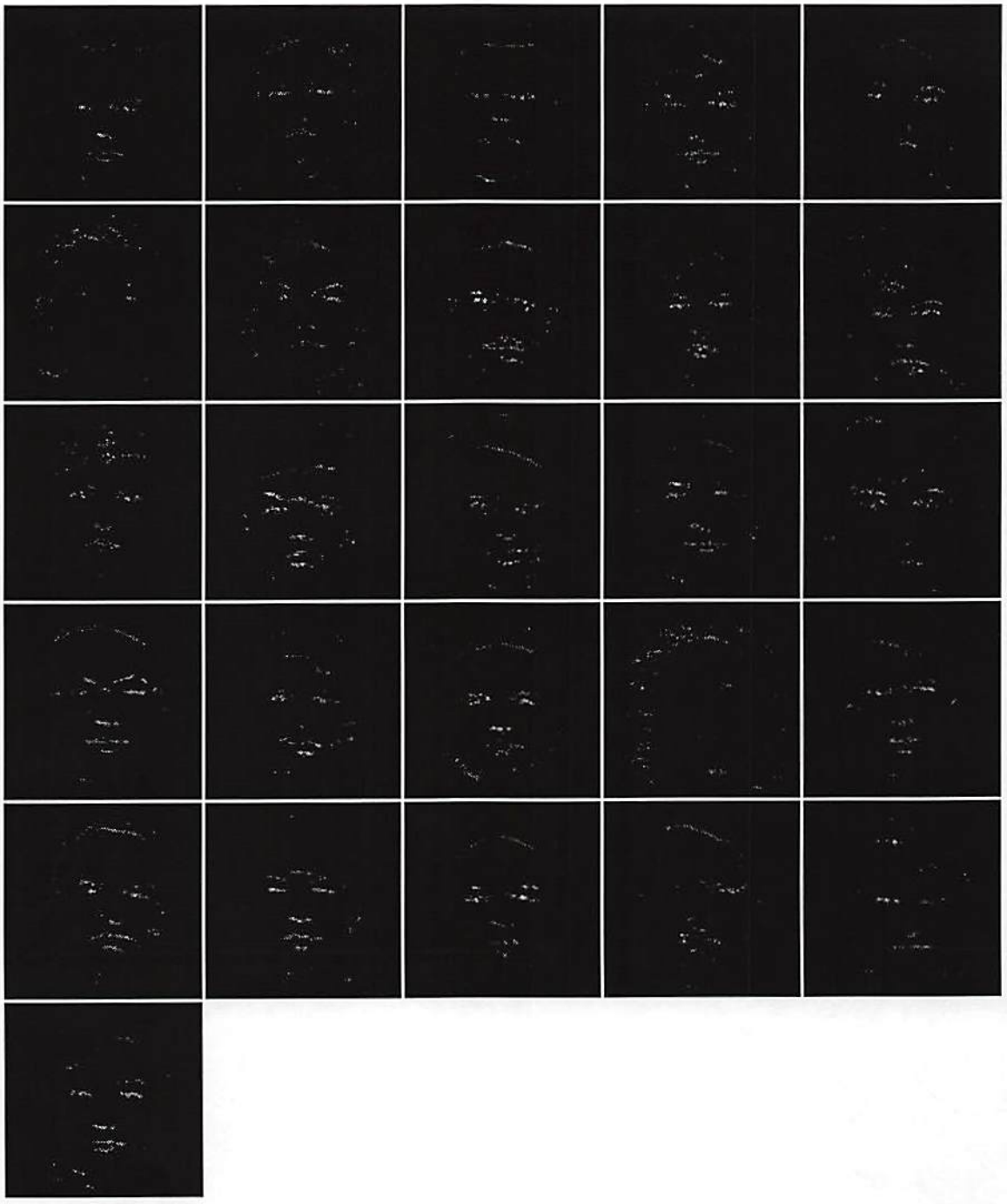


Figure 6.18: SVED neural network output (binary images, no post-processing).

binary photonic signal-shifting chip, but develop more sophisticated methods of image binarization, such as half-toning (*e.g.*, [Knuth, 1987]) or histogram-based techniques [Debenham and Garth, 1992].

For future work, it would be interesting to experiment with more general types of space variant pre-processing. In this dissertation we concentrated on space variant edge detection, but the pre-processing could also be based on a wavelet or Fourier decomposition, for example. These alternatives may make the object detection system more general than the "concentric rectangle" assumption inherent to the space-variant edge detection layer used in our system.

Chapter 7

Conclusions

This dissertation has discussed three classes of photonic neural-network models: (1) photonic neural networks based upon probability density estimation, (2) photorefractive neural-network models, and (3) vertically stacked photonic neural networks that utilize hybridized CMOS/GaAs chips and diffractive optical elements. In each case, we showed how previously developed neural-network learning algorithms and/or architectures must be adapted in order to allow an efficient photonic implementation. For class (1), we showed that conventional "k-Nearest Neighbors" (k-NN) density estimation is not suitable for an analog photonic neural-network hardware implementation, and we introduced a new density estimation algorithm called "Continuous k-Nearest Neighbors" (C-kNN) that is suitable. For class (2), we showed that the diffraction-efficiency decay inherent to photorefractive grating formation adversely affects outer-product neural-network learning algorithms, and we introduced a gain and exposure scheduling technique that resolves the incompatibility. For class (3), the use of compact diffractive optical interconnections constrains the corresponding neural-network weights to

be fixed and locally connected. We introduced a 3-D Photonic Multichip-Module (3-D PMCM) neural-network architecture that utilizes a fixed diffractive optical layer in conjunction with a programmable electronic layer, to obtain a multi-layer neural network capable of real-time pattern recognition tasks such as locating the eyes in an image of a human face.

In the discussion of class (1), we first introduced the C-kNN density estimation algorithm from an analytical viewpoint. A proof of statistical consistency for the new algorithm was given, along with empirical simulation results that suggest performance improvements in comparison to the conventional k-NN and Parzen Windows techniques. The remainder of Chapter 2 focused on neural network implementations and applications for C-kNN density estimation. Radial Basis Function (RBF) architectures were presented for C-kNN density estimation, Bayesian pattern classification, and nonparametric regression applications. Empirical simulation results showed better pattern classification performance when the density estimate was based on C-kNN than when the density estimate was based on Parzen Windows (the difference in performance was not statistically significant when compared to k-NN). As a potentially useful extension of this work, we suggested that it was possible to apply the C-kNN concept to a more general class of RBF neural networks; standard clustering algorithms could be used to determine the first layer of weights, but the basic concept of the kernel width being dependent on the input through a feedback mechanism could still be applied.

The discussion of class (1) continued in the next chapter, with the presentation of photonic architectures for neural networks based upon probability density estimation. Volume and planar holographic architectures for RBF neural-network

implementations of probability density estimation, Bayesian pattern classification, and nonparametric regression were presented. Photonic architectures for both C-kNN and Parzen Windows techniques were given, including a derivation of the relationship between Spatial Light Modulator (SLM) gain and estimation kernel width. These photonic architectures allow the computation of all the required square-distance calculations in parallel, thereby allowing the possibility of real-time pattern classification even when the training set is large.

The next chapter presented an analysis of neural-network class (2). We derived a gain and exposure scheduling technique that compensates for photorefractive diffraction-efficiency decay when applied to outer-product neural-network learning algorithms. Equations that map the photorefractive grating update equations into neural-network weight updates were presented, along with empirical simulation results which verify that our method works correctly when applied to the Exclusive-OR (XOR) problem. It was shown that the method applies to both Single-Coherent-Source (SCS) and Incoherent/Coherent (I/C) optical representations.

The final two technical chapters of this dissertation discussed neural-network class (3). The 3-D PMCM neural-network hardware architecture was presented first. In the hardware architecture chapter, we presented the VLSI design, computer simulation results, and electrical test results for the "photonic signal-shifting chip" that provides a scrolling-window input to the neural-network processor. Derivations for the mapping between neural-network signals and 3-D PMCM optical representations were given. The 3-D PMCM diffractive optical interconnections system was also discussed. We presented examples of fabricated micro-diffractive optical elements designed for neural-network edge detection, and we

derived optical imaging system constraints which showed that the 3-D PMCM optical components can be stacked compactly. Several potential extensions of this work were discussed, including the use of Vertical Cavity Surface Emitting Laser (VCSEL) arrays in place of the optical power bus/optical modulator combination, as well as the design of a CCD-version of the photonic signal-shifting chip to allow a gray-scale input plane.

In the last technical chapter of this dissertation, a two-layer neural network that is consistent with the 3-D PMCM hardware architecture was simulated for the application of detecting the positions of the two eyes in an image of a human face. Space-variant edge detection, which requires only local neural-network connectivity, was shown to be a useful pre-processing stage for eye detection. Simulation results on a database of real facial images showed significant performance improvements compared to a simpler single-layer network. We showed that the two-layer neural-network algorithm is computationally intensive when implemented on a digital computer: it took over 20 seconds to process a single image using a SPARC 10 workstation, while the 3-D PMCM neural network can potentially process the images in real time (1/30 second per image). As potential future work, we discussed the possibility of using the diffractive optical interconnection layer to implement a more general wavelet or Fourier decomposition.

Appendix A

Interpretation of the weighted-least-squares diagonal matrix

In Sec. 6.3.3, the weighted-least-squares expression [Mendel, 1987]

$$\hat{\mathbf{w}} = (\mathbf{M'}^T\mathbf{D}\mathbf{M'})^{-1}\mathbf{M'}^T\mathbf{D}\mathbf{t}, \tag{A.1}$$

given a diagonal weighting matrix $\mathbf{D}$, was introduced as the solution for the linear set of equations

$$\mathbf{M'}\mathbf{w} = \mathbf{t}. \tag{A.2}$$

A useful interpretation of the $\mathbf{D}$ matrix is derived in this appendix.

The weighted-least-squares solution given in Eq. (A.1) minimizes the weighted-squared-error J [Mendel, 1987], where J is defined according to

$$J(\hat{\mathbf{w}}) = (\mathbf{t} - \hat{\mathbf{t}})^T\mathbf{D}(\mathbf{t} - \hat{\mathbf{t}}), \tag{A.3}$$

in which the estimated output $\hat{\mathbf{t}}$ is given by

$$\hat{\mathbf{t}} = \mathbf{M}'\hat{\mathbf{w}}. \tag{A.4}$$

Because $\mathbf{D}$ is a diagonal matrix, Eq. (A.3) can be rewritten in the form

$$J(\hat{\mathbf{w}}) = \sum_{i=1}^{N} d_i(t_i - \hat{t}_i)^2, \tag{A.5}$$

where the weighting element d_i corresponds to component i of the error $(t_i - \hat{t}_i)$, and N is the dimensionality of the target vector $\mathbf{t}$. In the following, we assume that $\{d_i\}$ are integers.

One way to interpret the $\mathbf{D}$ matrix is to note that Eq. (A.1) is equivalent to the unweighted least-squares solution

$$\hat{\mathbf{w}} = (\mathbf{M}'^{(\mathbf{new})^T}\mathbf{M}'^{(\mathbf{new})})^{-1}\mathbf{M}'^{(\mathbf{new})^T}\mathbf{t}^{(\mathbf{new})}, \tag{A.6}$$

in which $\mathbf{M}'^{(\mathbf{new})}$ and $\mathbf{t}^{(\mathbf{new})}$ are defined by replicating each row i in $\mathbf{M}'$ and $\mathbf{t}$ into d_i identical rows. In so doing, the dimensionalities of $\mathbf{M}'^{(\mathbf{new})}$ and $\mathbf{t}'^{(\mathbf{new})}$ grow to size $\sum_i d_i$. The equivalence of Eq. (A.6) and Eq. (A.1) is shown by noting that Eq. (A.6) solves for the weight vector $\hat{\mathbf{w}}$ that minimizes the unweighted least-squares error,

$$J^{(new)}(\hat{\mathbf{w}}) = \sum_{i=1}^{(\sum_j d_j)} (t_i^{(new)} - \hat{t}_i^{(new)})^2. \tag{A.7}$$

However, $J^{(new)}(\hat{\mathbf{w}})$ must be equal to $J(\hat{\mathbf{w}})$, because the d_i repeated additions of replicated squared-error $(t_i^{(new)} - \hat{t}_i^{(new)})^2$ in Eq. (A.7) has the same effect as

the multiplicative factor d_i in Eq. (A.5). Therefore, the solution $\hat{\mathbf{w}}$ that minimizes Eq. (A.5) will be identical to that which minimizes Eq. (A.7), and likewise Eq. (A.1) yields the same solution as Eq. (A.6).

Appendix B

I/C weight update convergence proof

The neural interconnection weight update for the I/C architecture is given by

$$\begin{aligned} w_{ik}(n+1) &= w_{ik}(n) + (\alpha^2/4)\delta_i(n+1)x_k(n+1) \\ &+ \alpha\left(\sqrt{\delta_i^+(n+1)w_{ik}^+(n)} - \sqrt{\delta_i^-(n+1)w_{ik}^-(n)}\right)\sqrt{x_k(n+1)}, \end{aligned} \quad \text{(B.1)}$$

in which δ_i^+ is the positive component of δ_i $[\delta_i^+ = (1/2)(\delta_i + |\delta_i|)]$ and δ_i^- is the negative component of δ_i $[\delta_i^- = (1/2)(\delta_i - |\delta_i|)]$. The two unipolar weight components w_{ik}^+ and w_{ik}^- are related to the bipolar weight w_{ik} through the relation $w_{ik} = w_{ik}^+ - w_{ik}^-$. Clearly, Eq. (B.1) is not equivalent to the usual outer-product neural update given by

$$w_{ik}(n+1) = w_{ik}(n) + \alpha\delta_i(n+1)x_k(n+1). \quad \text{(B.2)}$$

It might seem surprising, therefore, that gradient descent neural network algorithms that were designed to converge using Eq. (B.2) will still converge using Eq. (B.1).

Gradient descent algorithms will still converge using Eq. (B.1) because the *sign* of the weight-change $(w_{ik}(n+1) - w_{ik}(n))$ is the same as that given by the standard outer-product rule of Eq. (B.2). The sign of the I/C weight-change shown in Eq. (B.1) is equal to the sign of $\delta_i(n+1)$, because the input signals $x_k(n+1)$ are assumed to be positive, as are the unipolar weight components $w_{ik}^{+}(n)$ and $w_{ik}^{-}(n)$. Similarly, the sign of the standard weight-change shown in Eq. (B.2) is also equal to the sign of $\delta_i(n+1)$.

The fact that both the weight-changes have the same sign is important for the following reason. All gradient descent neural network algorithms seek to minimize an error function by altering the interconnection weights. The change in error function J given a set of weight-changes $\{w_{ik}\}$ is approximately equal to

$$\Delta J = \sum_{ik} \Delta w_{ik} \frac{\partial J}{\partial w_{ik}}, \tag{B.3}$$

in which small weight updates are assumed. The above equation shows that when the weight updates are chosen to be in a direction opposite to the error gradient (*i.e.* gradient descent) the decrease in error is maximized. However, it is not necessary to implement a true gradient descent in order to guarantee network convergence. It is only necessary to show that the sign of the RHS of Eq. (B.3) is negative at each iteration. If the sign of Δw_{ik} is the same as that of the gradient descent weight-change $-\frac{\partial J}{\partial w_{ik}}$, then Eq.(B.3) shows that ΔJ must be negative. Therefore, if the standard update given by Eq. (B.2) implements a gradient descent of any error function J, then any other update rule that yields weight-changes of the same sign will always decrease J (but the decrease will not be in the optimal gradient descent direction). This implies that the I/C weight

update must decrease the network error at each iteration, because the sign of the weight-change given in Eq. (B.1) is the same as that of the gradient descent weight-change given in Eq. (B.2).

Appendix C

Proportional control for C-kNN feedback

An analog proportional controller [Kailath, 1980] can be used to determine the kernel width $h_n(\mathbf{x})$ for an RBF neural network implementation of C-kNN density estimation, under the constraint that the restricted kernel function $\phi_r(\cdot)$ is a monotonically decreasing function of its input argument:

$$\phi_r\prime(p) < 0 \quad \forall p > 0. \tag{C.1}$$

We prove the above assertion as follows.

For the C-kNN density estimation application, a proportional controller can be defined by the equation (see Fig. 2.6),

$$\frac{dh_n(\mathbf{x})}{dt} = \alpha[k_n^{(C)} - S(\mathbf{x})], \tag{C.2}$$

where

$$S(\mathbf{x}) = \sum_{i=1}^{n} \phi\left(\frac{\mathbf{x} - \mathbf{t}^{(i)}}{h_n(\mathbf{x})}\right) \tag{C.3}$$

and α is a positive constant. Convergence to the correct kernel width is shown

using a Lyapunov error function defined by

$$J = [S(\mathbf{x}) - k_n^{(C)}]^2. \tag{C.4}$$

Taking the time-derivative of Eq. (C.4) yields

$$\frac{dJ}{dt} = 2[S(\mathbf{x}) - k_n^{(C)}] \left[\sum_{i=1}^{n} \phi\prime \left(\frac{\mathbf{x} - \mathbf{t}^{(i)}}{h_n(\mathbf{x})}\right)\right] (-h_n(\mathbf{x})^2) \left(\frac{dh_n(\mathbf{x})}{dt}\right). \tag{C.5}$$

Substitution of Eq. (C.2) into Eq. (C.5), while noting that the summation term in Eq. (C.5) is negative due to Eq. (C.1) and that $h_n(\mathbf{x})$ is always positive, shows that $\frac{dJ}{dt}$ is a negative quantity. Therefore, because J is non-negative by definition, the steady state solution to Eqs. (C.2) and (C.3) must satisfy the constraint $S(\mathbf{x}) = k_n^{(C)}$ □.

Appendix D

Equivalence of voting C-kNN and volumetric C-kNN classification rules

For the $N = 2$ case, the voting C-kNN rule using parameter $k_n^{(C)}$ is equivalent to the volumetric C-kNN rule using parameter $k_n^{(C)}/2$, when $\phi(\mathbf{x})$ is continuous. This relationship is proved as follows. We assume without loss of generality that the voting C-kNN rule chooses class ω_1 for a given input vector $\mathbf{x}$ and smoothing parameter $k_n^{(C)}$, and we show that the volumetric C-kNN rule will also choose class ω_1 when smoothing parameter $k_n^{(C)}/2$ is used.

If the voting rule chooses ω_1, then by definition $Z(\omega_1|\mathbf{x}) > Z(\omega_2|\mathbf{x})$. Using Eq. (2.28), it is clear that the relations

$$Z(\omega_1|\mathbf{x}) > k_n^{(C)}/2 \tag{D.1}$$

$$Z(\omega_2|\mathbf{x}) < k_n^{(C)}/2 \tag{D.2}$$

must hold. If the value of $h_n(\mathbf{x})$ determined by the voting rule in Eq. (2.28) is denoted h_{vote}, and the two values of $h_n(\mathbf{x})$ obtained for the volumetric rule using smoothing parameter $k_n^{(C)}/2$ are denoted h_1 (for class ω_1) and h_2 (for class ω_2),

then it must be true that $h_1 < h_{\text{vote}}$ and $h_2 > h_{\text{vote}}$. This is because the RHS of Eq. (2.27) is identical to the LHS of Eq. (2.26). During the volumetric C-kNN procedure, if the kernel width starts out as $h_n(\mathbf{x}) = h_{\text{vote}}$, then $h_n(\mathbf{x})$ must decrease from its initial value to arrive at its final value h_1 because the LHS of Eq. (2.26) applied to class ω_1 is larger than $k_n^{(C)}/2$, due to Eq. (D.1). Similarly, for class ω_2, $h_n(\mathbf{x})$ must increase from h_{vote} to its final value h_2. Therefore, $h_1 < h_2$, and the volumetric C-kNN classification rule also chooses class ω_1 □.

Appendix E

Effect of substrate thickness and/or wavelength on optical wavefront propagation

When designing diffractive optical elements (DOE's) for complicated optical systems, it is frequently useful to neglect the thickness and refractive index of the DOE substrate(s), and to assume a known wavelength of operation. In the actual optical system, however, substrate thicknesses may have a significant effect on the propagation of the optical wavefront, and it also may be necessary to alter the operating wavelength. This appendix describes a simple method that allows an optical engineer to design diffractive elements such as microlenses, microDOE's, or apertures without needing to fix the wavelength and/or substrate thicknesses before specifying the required transmittance functions.

An example of a simplified diffractive optical system in which the substrate is ignored is given in Fig. E.1 (a). In the figure, a diffractive microlens that was designed to operate at wavelength λ is shown. (It should be noted that the following derivation applies to all diffractive optical elements, not just to microlenses.) A planar wavefront is transformed by the thin diffractive element into a converging beam with complex amplitude $u(x; z = 0)$ at the plane $z = 0$.

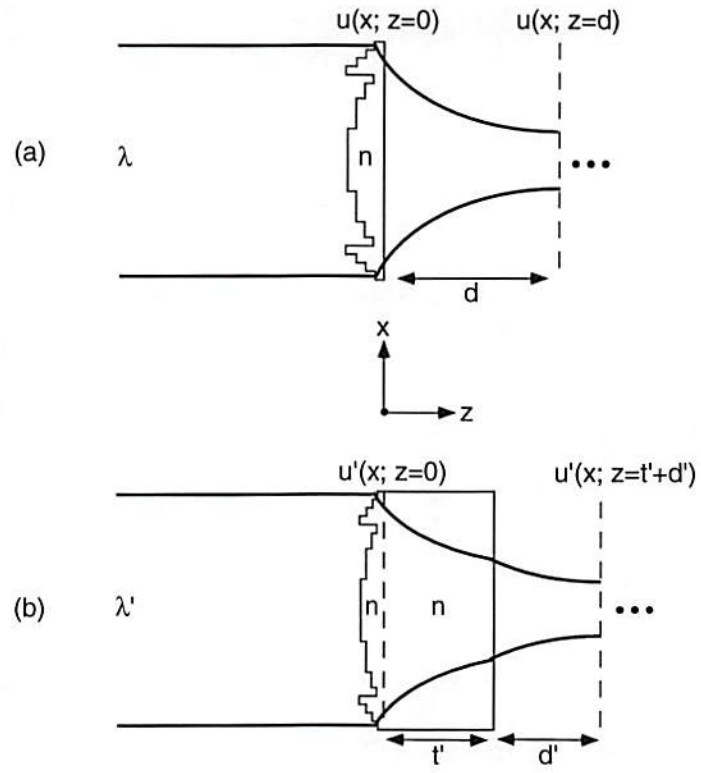


Figure E.1: Compensation for changes in refractive index and wavelength. In part (a), a thin DOE with no substrate is assumed. In part (b), the wavelength is altered and a finite substrate is present.

After propagating a distance d through free space, the complex amplitude becomes $u(x; z = d)$.

In Fig. E.1 (b), the operating wavelength has been changed to λ', and a substrate with thickness t' with an index of refraction equal to n is present. In the following, we show that the same microlens design used for Fig. E.1 (a) can be applied to the new optical system, under the condition that the propagation distance and etch depth are properly chosen. If a new spacing d' and a new etch depth are chosen such that

$$u'(x; z = d' + t') = u(x; z = d), \tag{E.1}$$

in which u' is the new complex amplitude and t' represents the substrate thickness, then the final complex amplitude of the optical wavefront shown in part (b) is the same as that originally designed for the simpler system shown in part (a). The optical wavefront will then propagate to any subsequent components in the optical system as originally designed.

The propagation of an optical wavefront through free space [part (a) of the figure] can be expressed in the Fourier domain as [Goodman, 1996]

$$U(f; z = d) = U(f; z = 0) \exp[-j\pi f^2 \lambda d], \tag{E.2}$$

in which $U(f; z)$ is the Fourier transform of $u(x; z)$, j is $\sqrt{-1}$, and a constant phase factor that does not influence wavefront propagation has been ignored. This expression represents the Fresnel approximation to free-space propagation, and therefore it only applies in the paraxial limit [Goodman, 1996]. The two wavefront propagations shown in part (b) of the figure can be combined using the expression

$$U'(f; z = t' + d') = U'(f; z = 0) \exp[-j\pi f^2 (\lambda' t'/n + \lambda' d')]. \tag{E.3}$$

If the Fourier-space equality $U'(f; z = t' + d') = U(f; z = d)$ is satisfied, then the spatial-domain equality given in Eq. (E.1) must also be satisfied. A comparison between Eqs. (E.2) and (E.3) shows that the distance d' should be chosen such that

$$\lambda' t'/n + \lambda' d' = \lambda d, \tag{E.4}$$

or

$$d' = (\lambda/\lambda')d - t'/n. \tag{E.5}$$

A comparison between Eqs. (E.2) and (E.3) also shows that the Fourier-domain equality $U'(f; z = 0) = U(f; z = 0)$ must be satisfied, or, equivalently,

$$u'(x; z = 0) = u(x; z = 0). \tag{E.6}$$

This equality between the two complex amplitudes will be satisfied if the phase-only transmittance functions for the two microlenses are the same. Because the phase-only transmittance function of a diffractive element is proportional to the ratio between its point-wise thickness and the wavelength of operation, Eq. (E.6) is satisfied if the etch depth for the system shown in part (b) is scaled by the wavelength ratio λ'/λ.

To summarize, diffractive optical elements can be designed without needing to fix the substrate thickness and/or wavelength of operation *a priori*. Provided that the spacing between optical components is adjusted according to Eq. (E.4), and the etch depth is scaled in proportion to the ratio between the new and original wavelengths, the DOE design is independent of any changes in substrate thickness and/or wavelength.

References

[Ananthanarayanan *et al.*, 1995] K. Ananthanarayanan, C.-H. Chen, S. DeMars, A. A. Goldstein, C. C. Huang, D. Su, C. B. Kuznia, C. Kyriakakis, Z. Karim, B. K. Jenkins, A. A. Sawchuk, A. R. Tanguay, Jr., "Multilayer Electronic/Photonic Multichip Modules with Vertical Optical Interconnections", presented at *OSA annual meeting* (Portland, Oregon, 9/10/95 - 9/15/95).

[Anderson, 1987] D. Z. Anderson, "Dynamic optical interconnects: volume holograms as optical two-port operators", *Applied Optics* **26**, 5031 (1987).

[Anderson and Erie, 1987] D. Z. Anderson, M. C. Erie, "Resonator memories and optical novelty filters", *Optical Engineering* **26**, 434-444 (1987).

[Asthana *et al.*, 1993] P. Asthana, G. P. Nordin, A. R. Tanguay, Jr., and B. K. Jenkins, "Analysis of weighted fan-out/fan-in volume holographic optical interconnections,", *Applied Optics* **32**, 1441-1469 (1993).

[Asthana *et al.*, 1990] P. Asthana, H. Chin, G. Nordin, A. R. Tanguay, Jr., G. C. Petrisor, B. K. Jenkins, A. Madhukar, "Photonic components for neural net implementations using incoherent/coherent holographic interconnections" *Technical Digest: OSA Annual Meeting* (Boston, Massachusetts, November 1990).

[Baily and Jain, 1978] T. Bailey, A. K. Jain, "A note on distance-weighted k-Nearest Neighbor rule", *IEEE Transactions on Systems, Man, and Cybernetics* **8** (4), 311-313 (1978).

[Bellman, 1961] R. E. Bellman, *Adaptive Control Processes* (Princeton University Press, Princeton N.J., 1961).

[Benlarbi and Solymar, 1978] B. Benlarbi, L. Solymar, "The effect of the relative intensity of the reference beam upon the reconstructing properties of volume phase holograms", *Optica Acta* **26**, 271-278 (1978).

[Bezdek *et al.*, 1986] J. C. Bezdek, S. K. Chuali, D. Leep, "Generalized k-NN rules", *Fuzzy Sets and Systems* **1**, 237-256 (1986).

[Blotekjaer, 1979] K. Blotekjaer, "Limitations on holographic storage capacity of photochromic and photorefractive media", *Applied Optics* **18**, 57 (1979).

[Brady *et al.*, 1990] D. Brady, K. Hsu, D. Psaltis, "Periodically refreshed multiply exposed photorefractive holograms", *Optics Letters* **15**, 817 (1990).

[Buturovic, 1993] L. J. Buturovic, "Improving k-Nearest Neighbors Density and Error Estimates", *Pattern Recognition* **26** (4), 611-616 (1993).

[Cacoullos, 1964] T. Cacoullos, "Estimation of a multivariate density", Technical Report No. 40, Department of Statistics, University of Minnesota (1964).

[Casasent, 1984] D. Casasent, "Unified synthetic discriminant function computational formulation", *Applied Optics* **25** 2343-2350 (1984).

[Caulfield, 1987] H. J. Caulfield, "Parallel N^4 Weighted Optical Interconnects," *Applied Optics* **26**, 4039 - 4040 (1987).

[Chiang and Chuang, 1991] A. M. Chiang, M. L. Chuang, "A CCD programmable image processor and its neural network applications", *IEEE Journal of Solid-State Circuits* **26** (12), 1894-1901 (1991).

[Dasarathy, 1991] B. V. Dasarathy, *NN Pattern Classification Techniques* (IEEE computer society press, Los Alamitos, CA, 1991).

[Debenham and Garth, 1992] R. M. Debenham, S. C. J. Garth, "The detection of eyes in facial images using radial basis functions", in *Neural Networks for Vision, Speech, and Natural Language Processing* (Edited by Linggard R., *et al.*, 1992), pp. 64-92.

[Demars, 1997] S. Demars, Ph. D. Thesis, University of Southern California (1997).

[Denker *et al.*, 1989] J. S. Denker, W. R. Gardner, H. P. Graf, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, H. S. Baird, I. Guyon, "Neural Network Recognizer for Handwritten Zip Code Digits", *Neural Information Processing Systems* **1**, 323-331 (1989).

[Denoeux, 1995] T. Denoeux, "A k-Nearest Neighbor Classification Rule Based on Dempster-Shafer Theory", *IEEE Transactions on Systems, Man, and Cybernetics* **25** (5), 804-813 (1995).

[Duda and Hart, 1973] R. E. Duda, P. E. Hart, *Pattern recognition and scene Analysis* (Wiley, New York, 1973), Ch. 4.

[Dudani, 1976] S. A. Dudani, "The Distance-Weighted k-Nearest-Neighbor Rule", *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-6**, 325-327 (1976).

[Fan *et al.*, 1995] J. Fan, D. Zaleta, C. K. Cheng, S. H. Lee, "Physical models and algorithms for optoelectronic MCM layout", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **3** (1), pp. 124-135 (1995).

[Farhat *et al.*, 1985] N. H. Farhat, D. Psaltis, A. Prata, "Optical implementation of the Hopfield model", *Applied Optics* **24**, 1469-1475 (1985).

[Feldman *et al.*, 1994] M. R. Feldman, J. E. Morris, I Turlik, P. Magill, G. Adema, M. Y. A. Raja, "Holographic optical interconnects for VLSI multichip modules", *IEEE Transactions on Components Packaging and Manufacturing Technology Part B - Advanced Packaging* **17** (2), pp. 223-227 (1994).

[Feldman *et al.*, 1988] M. R. Feldman, S. C. Esener, C. C. Guest, and S. H. Lee, "Comparison between optical and electrical interconnects based on power and speed considerations", *Applied Optics* **27**, pp. 1742-1751 (1988).

[Fellous, 1996] J. M. Fellous, Ph. D. dissertation, University of Southern California (1996).

[Fix and Hodges, 1951] E. Fix, J. L. Hodges, "Discriminatory analysis, nonparametric discrimination, consistency properties", Rept. No. 4, Project 21-49-004, School of Aviation Medicine, Randoloph Field, Texas (1951).

[Fukunaga, 1990] Fukunaga, *Statistical pattern recognition* (Academic Press, Boston, 1990).

[Garvin, 1995] H. L. Garvin, "Three-dimensional packaging", in *Multichip module design, fabrication, and testing*, by J. J. Licari, (New York, McGraw-Hill, 1995).

[Geman *et al.*, 1992] S. Geman, E. Bienenstock, R. Doursat, "Neural Networks and the Bias/Variance Dilemma", *Neural Computation*, **4**, 1-58 (1992).

[Goldstein and Jenkins, 1992] A. A. Goldstein, B. K. Jenkins, "Optical probability density estimation for real-time pattern classification", *OSA Annual Meeting Technical Digest* (OSA, Washington D.C., 1992), paper ThQ3.

[Goldstein *et al.*, 1995] A. A. Goldstein, G. C. Petrisor, B. K. Jenkins, "Gain and exposure scheduling to compensate for photorefractive neural-network weight decay", *Optics Letters* **20**, 611-613 (1995).

[Goodman, 1996] J. W. Goodman, *Introduction to Fourier Optics* (McGraw-Hill, New York, 1996).

[Goodman, 1984] J. W. Goodman, "Optical interconnections for VLSI systems", *Proceedings of the IEEE* **72**, 850-866 (1984).

[Goossen *et al.*, 1995] K. W. Goossen, A. L. Lentine, J. A. Walker, L. A. D'Asaro, S. P. Hui, B. Tseng, R. Leibenguth, D. Kossives, D. Dahringer, L.M.F. Chirovsky, and D. A. B. Miller, "Demonstration of a dense, high-speed optoelectronic technology integrated with silicon CMOS via flip-chip bonding and substrate removal", *Proceedings of OC' 95* (Salt Lake City, 1995), paper OTuC1-1.

[Gray and Meyer, 1993] Paul R. Gray, Robert G. Meyer, *Analysis and design of analog integrated circuits* (Wiley, New York, 1993).

[Hegelin and Hewit, 1994] P. M. Hagelin, J. R. Hewit, "Artificial Neural Networks for Locating Eyes in Facial Images", *Mechatronics* **4** (7), 737-752 (1994).

[Hardle, 1990] W. Hardle, *Applied nonparametric regression* (Cambridge University Press, New York, 1990), Ch. 3.

[Hinton *et al.*, 1994] H. S. Hinton, T. J. Cloonan, F. B. McCormick, A. L Lentine, F. A. P. Tooley, "Free-space digital optical systems", *Proceedings of the IEEE* **81** (11), 1632-1649 (1994).

[Hong, *et al.*, 1990] J. H. Hong, S. Campbell, P. Yeh, "Optical classifier with perceptron learning", *Applied Optics* **29**, 3019-3025 (1990).

[Hopfield, 1983] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", *Proceedings of the National Academy of Sciences USA* **79** 2554-2558 (1983).

[Hornik *et al.*, 1990] K. Hornik, M. Stinchcombe, H. White, "Multilayer feedforward networks are universal approximators", *Neural Networks* **2**, 359-366 (1990).

[Hsu *et al.*, 1993] K. Y. Hsu, S. H. Lin, P. Yeh, "Conditional convergence of photorefractive perceptron learning", *Optics Letters* **18**, 2135 (1993).

[Huang, *et al.*, 1997] C.-C. Huang, B. K. Jenkins, C. B. Kuznia, "Space-variant interconnections based on diffractive optical elements for neural networks: architectures and crosstalk", to be submitted to *Applied Optics*, Spring 1997.

[Hwang *et al.*, 1994] J. Hwang, S. Lay, M. Maechler, R. D. Martin, "Regression

modeling in Back-propagation and Projection pursuit learning", *IEEE Transactions on Neural Networks* **5**, 342-353 (1994).

[Jacob, 1993] Robert J. K. Jacob, "Eye-Gaze Computer Interfaces", *Computer* **26** (7), 65-66 (1993).

[Jang *et al.*, 1988] J. Jang, S. Jung, S. Lee, S. Shin, "Optical implementation of the Hopfield model for two-dimensional associative memory", *Optics Letters* **13**, 248 - 250 (1988).

[Jahns, 1994] J. Jahns, "Diffractive optical elements for optical computers", in *Optical Computing Hardware* (Academic Press, San Diego, CA, 1994), Ch. 6.

[Jahns, 1994] J. Jahns, "Planar packaging of free-space optical interconnections", *Proceedings of the IEEE* **82** (11), pp. 1623-1631 (1994).

[Jenkins *et al.*, 1990] B. K. Jenkins, G. C. Petrisor, S. Piazzolla, P. Asthana, and A. R. Tanguay, Jr. "Photonic architecture for neural nets using incoherent/coherent holographic interconnections," in *Optical Computing '90*, J. Tsujinunchi, Y. Ichioka, and S. Ishihara, eds., Proc. Soc. Photo-Opt. Instrum. Eng. **1359**, 317-318 (1990).

[Jenkins and Tanguay, 1992] B. K. Jenkins, A. R. Tanguay Jr., "Photonic Implementations of Neural Networks", in B. Kosko ed., *Neural Networks for Signal Processing* (Prentice Hall, Inglewood Cliff N.J., 1992).

[Johnson *et al.*, 1984] K. M. Johnson, H. Lambertus, J. W. Goodman, "Holographic reciprocity law failure", *Applied Optics* **23**, 218-227 (1984).

[Johnson, 1996] R. Johnson, "PC video system simplifies security", *Electronic Engineering Times*, n8, 1996.

[Keller *et al.*, 1985] J. M. Keller, M. R. Gray, J. A. Givens, "A fuzzy k-Nearest Neighbor Algorithm", *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-15**, 580-585 (1985).

[Kailath, 1980] T. Kailath, *Linear Systems* (Englewood Cliffs, N.J., 1980).

[Kissiov and Hadjitodorov, 1992] V. T. Kissiov, S. T. Hadjitodorov, "A fuzzy version of the k-NN method", *Fuzzy Sets and Systems* **49**, 323-329 (1992).

[Kosko, 1992] B. Kosko, *Neural networks and fuzzy Systems: a dynamical systems approach to machine intelligence* (Prentice Hall, Englewood Ciffs, NJ, 1992).

[Krishnamoorthy *et al.*, 1996] A. V. Krishnamoorthy, J. E. Ford, K. W. Goossen, J. A. Walker, A. L. Lentine, S. P. Hui, B. Tseng, L. M. F. Chirovsky, R.

Leibenguth, D. Kossives, D. Dahringer, L. A. D'Asaro, F. E. Kiamilev, G. F. Aplin, R. G. Rozier, D. A. B. Miller, "Photonic page buffer based on GaAs multiple-quantum-well modulators bonded directly over active silicon complementary-metal-oxide-semiconductor (CMOS) circuits", *Applied Optics* **35** (3), 2439-2448 (1996).

[Krueger *et al.*, 1996] N. Krueger, M. Poetzch, C. von der Malsburg, "Determination of face position and pose with a learned representation based on labeled graphs" (Internal Report No. 96-03), Universitat Bochum, Institut fur Neuroinformatik, 44780 Bochum, Germany, 1996.

[Knuth, 1987] P. E. Knuth, "Digital Half-tones by Dot Diffusion", *ACM Transactions on Graphics* **6** (4), pp. 245-253 (1987).

[Kuznia *et al.*, 1995] C. B. Kuznia, C. C. Huang, K. Ananthanarayanan, C.-H. Chen, A. A. Sawchuk, "Micro-diffractive optical elements for smart pixel fanout interconnections", *OSA Annual Meeting* (Portland, 1995), paper ThCC3.

[Kyriakakis *et al.*, 1995] C. Kyriakakis, Z. Karim, A. R. Tanguay, Jr., R. F. Cartland, A. Madhukar, S. Piazolla, B. K. Jenkins, C. B. Kuznia, A. A. Sawchuk, C. von der Malsburg, "Photonic implementations of neural networks", in *Optical Computing*, Vol. 10 of 1995 OSA Technical Digest Series (Optical Society of America, Washington, D. C., 1995), pp. 128-130.

[Li *et al.*, 1993] H. Y. Li, Y. Qiao, D. Psaltis, "Optical network for real-time face recognition", *Applied Optics* **32**, 5026-5035 (1993).

[Licari, 1995] J. J. Licari, *Multichip module design, fabrication, and testing* (New York, McGraw-Hill, 1995).

[Lin, 1990] F. Lin, "Practical realizations of N^4 optical interconnects", *Applied Optics* **29**, 5226-5227 (1990).

[Linggard *et al.*, 1992] R. Linggard, D. J. Myers, and C. Nightingale (ed.), *Neural Networks for Vision, Speech and Natural Language* (Chapman & Hall, 1992).

[Loftsgaarden and Quesenberry, 1965] D. O. Loftsgaarden, C. P. Quesenberry, "A nonparametric estimate of a multivariate density function", *Annals of Mathematical Statistics* **36**, 1049 - 1051 (1965).

[Lu and Lin, 1991] T. Lu, F. Lin, "Experimental Demonstration of Large-Scale Holographic Optical Neural Network", *IJCNN Proceedings* (1991).

[Lades *et al.*, 1993] M. Lades, J. C. Vorbruggen, J. Buhmann, J. Lange, C. v.d. Malsburg, R. P. Wurtz, W. Konen, "Distortion Invariant Object Recognition

in the Dynamic Link Architecture", *IEEE Transaction on Computers* **42**, 300-311 (1993).

[Macleod *et al.*, 1987] J. E. S. Macleod, A. Luk, D. M. Titterington, "A Re-examination of the distance-weighted k-Nearest Neighbor Rule", *IEEE Transactions on Systems, Man , and Cybernetics* **17**(4), 689-696 (1987).

[Mack and Rosenblatt, 1979] Y. P. Mack, M. Rosenblatt, "Multivariate k-Nearest Neighbor density estimates", *Journal of Multivariate Analysis* **9**, 1-15 (1979).

[Miller, 1989] D. Miller, "Optics for low-energy communication inside digital processors: quantum detectors, sources, and modulators as efficient impedance converters", *Optics Letters* **14**, 146-148 (1989).

[Mendel, 1987] Jerry M. Mendel, *Lessons in Digital Estimation Theory*, (Prentice-Hall Inc., 1987), Ch. 3.

[Moody and Darken, 1989] J. Moody, C. J. Darken, "Fast learning of locally-tuned processing units", *Neural Computation* **1**, 2281-294 (1989).

[Neifeld and Psaltis, 1993] M. A. Neifeld, D. Psaltis, "Optical implementations of radial basis classifiers", *Applied Optics* **32**, 1370-1379 (1993).

[Nishihara, 1989] H. Nishihara, *Optical integrated circuits* (McGraw-Hill, 1989).

[Owechko, 1993] Y. Owechko, "Cascaded-grating holography for artificial neural networks", *Applied Optics* **32**, 1380 (1993).

[Oikawa and Hamanaka, 1994] M. Oikawa, K. Hamanaka, "Physics of planar microlenses", in *Optical Computing Hardware* (Academic Press, San Diego, CA, 1994), Ch. 5.

[Paek and Psaltis, 1987] E. G. Paek, D. Psaltis, "Optical associative memory using Fourier transform holograms", *Optical Engineering* **26**, 428-433 (1987).

[Papoulis, 1965] A. Papoulis, *Probability, random variables, and stochastic processes*, Ch. 8 (1965).

[Parzen, 1962] E. Parzen, "On estimation of a probability density function and mode," *Annals of Mathematical Statistics* **33**, 1065-1076 (1962).

[Pentland *et al.*, 1994] Alex Pentland, Baback Moghaddam, Thad Starner, "View-based and modular eigenspaces for face recognition", in *Computer Vision and Pattern Recognition*, pages 84-91 (1994).

[Petrisor *et al.*, 1995] G. C. Petrisor, A. A. Goldstein, B. K. Jenkins, E. J. Herbulock, A. R. Tanguay, *OC '95 Technical Digest Series* (OSA, Washington D.C., 1995), paper OTuE1.

[Poggio, 1990] T. Poggio, "Networks for Approximation and Learning", *Proceedings of the IEEE* **78**, 1481-1497 (1990).

[Poggio and Beymer, 1996] T. Poggio, D. Beymer, "Learning to See", *IEEE Spectrum* **32** (5), 60-69 (1996).

[Pratt, 1991] W. K. Pratt, *Digital Image Processing* (John Wiley & Sons Inc., 1991).

[Psaltis *et al.*, 1988] D. Psaltis, D. Brady, K. Wagner, "Adaptive optical networks using photorefractive crystals", *Applied Optics* **27**, 1752 (1988).

[Psaltis and Hong, 1986] D. Psaltis, J. Hong, "Shift-invariant optical associative memories", Optical Engineering **26**, 10-15 (1986).

[Qiao *et al.*, 1991] Y. Qiao, D. Psaltis, C. Gu, J. H. Hong, "Phase-locked sustainment of photorefractive holograms using phase conjugation", *Journal of Applied Physics* **70**, 4646 (1991).

[Reilly *et al.*, 1982] D. L. Reilly, L. N. Cooper, C. Elbaum, "A neural model for category learning", *Biological Cybernetics* **45**, 35-41 (1982).

[Rogers *et al.*, 1995] S. K. Rogers, J. M Colombi, C. E. Martin, J. C. Gainey, K. H. Fielding, T. J. Burns, D. W. Ruck, M. Kabrisky, and M. Oxley, "Neural Networks for Automatic Target Recognition", *Neural Networks* **8** (7), 1153-1184 (1995).

[Rosenblatt, 1962] M. Rosenblatt, "Remarks on some nonparametric estimates of a density function", *Annals of Mathematical Statistics* **27**, 832-837 (1962).

[Rowley *et al.*, 1995] H. A. Rowley, S. Baluja, T. Kanade, "Human Face Detection in Visual Scenes" (Internal Report No. CMU-CS-95-158), Carnegie Mellon University (July 1995).

[Rumelhart and McClelland, 1986] D. Rumelhart, J. McClelland, *Parallel Distributed Processing* (MIT Press, Cambridge, MA, 1986), Ch. 8.

[San Jose Mercury News, 1996] "Iris-scanning technology for automatic teller machines" (Technology Information) *San Jose Mercury News* (May 28, 1996): p1E(2).

[Samal and Iyengar, 1992] A. Samal, P. A. Iyengar, "Automatic Recognition and Analysis of Human Faces and Facial Expressions: A Survey", *Pattern Recognition* bf 25, 65-77 (1992).

[Schioler and Hartmann, 1992] H. Schioler, U. Hartmann, "Mapping neural network derived frmo the Parzen window estimator", *Neural Networks'* **5** 903-909 (1992).

[Shamir *et al.*, 1989] J. Shamir, H. J. Caulfield, R. B. Johnson, "Massive holographic interconnection networks and their limitations", *Applied Optics* **28**, 311-324 (1989).

[Shavlik *et al.*, 1990] J. W. Shavlik, T. G. Dietterich, *Readings in Machine Learning* (Morgan-Kaufmann, Palo Alto CA, 1990).

[Sheu and Choi, 1995] Bing J. Sheu, Joongho Choi, *Neural information processing and VLSI* (Kluwer Academic Publishers, Norwell MA, 1995).

[Silverman, 1986] B. W. Silverman, *Density Estimation for Satistics and Data Analysis* (Chapman & Hall, New York, 1986), Ch. 3.

[Specht, 1990] D. F. Specht, "Probabilistic Neural Networks", *Neural Networks* **3**, 109-118 (1990).

[Specht, 1991] D. F. Specht, "A general regression neural network", *IEEE Transactions on Neural Networks* **2**, 568-576 (1991)

[Specht, 1993] D. F. Specht, "The general regression neural network - rediscovere d", *Neural Networks* **6** 1033-1034 (1993).

[Sung and Poggio, 1994] Kah-Kay Sung and Tomaso Poggio, "Example-based learning for view-based human face detection", *A.I. Memo CBCL Paper 112*, MIT, December 1994.

[Taketomi *et al.*, 1991] Y. Taketomi, J. E. Ford, H. Sasaki, J. Ma, Y. Fainman, S. H. Lee, "Incremental recording for photorefractive hologram multiplexing", *Optics Letters* **16** 1774-1776 (1991).

[Turunen *et al.*, 1990] J. Turunen, E. Tervonen, A. T. Friberg, "Acousto-optic control and modulation of optical coherence by electronically sysnthesized holographic gratings", *Journal of Applied Physics* **67**, 49-59 (1990).

[Urahama and Nagao, 1994] K. Urahama, T. Nagao, "Analog circuit implementation and learning algorithm for nearest-neighbor classifiers", *Pattern Recognition Letters* **15** 723-730 (1994).

[Veldkamp, 1993] B. W. Veldkamp, "Wireless focal planes on the road to amacronic sensors", *IEEE Journal of Quantum Electronics* **29**(2), pp. 801-813 (1993).

[van Heerden, 1963] P. J. Van Heerden, "Theory of optical information storage in solids", *Applied Optics* **2**(4), pp. 393-400 (1963).

[Waibel *et al.*, 1989] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, K. J. Lang, "Phoneme recognition using time-delay neural networks", *IEEE Transactions on Acoustics, Speech, and Signal Processing* **37** (3), 328-339 (1989).

[Werbos, 1974] P. J. Werbos, "Beyond regression: new tools for prediction and analysis in the behavioral sciences", Ph. D. Thesis, Harvard University, Cambridge, MA (1974).

[White, 1989] H. White, "Learning in artificial neural networks: A statistical perspective", *Neural Computation* **1**, 425-464 (1989).

[White, 1990] H. White, "Connectionist Nonparametric Regression: Multilayer Feedforward Networks Can Learn Arbitrary Mappings", *Neural Networks* **3**, 535-549 (1990).

[Wiley *et al.*, 1993] D. J. Wiley, I. Glaser, B. K. Jenkins, A. A. Sawchuk, "Incoherent dynamic lenslet array processor", *Applied Optics* **32**, 3641-3653 (1993).

[Wills *et al.*, 1995] D. S. Wills, W. S. Lacy, C. Camperiginestet, B. Buchanan, H. H. Cat, S. Wilkinson, M. H. Lee, N. M. Jokerst, M. A. Brooke, "A 3-dimensional high-throughput architecture using through-wafer optical interconnect", *Journal of Lightwave Technology* **13** (6), pp. 1085-1092 (1995).

[Worchesky *et al.*, 1996] T. L. Worchesky, K. J. Ritter, R. Martin, B. Lane, "Large arrays of spatial light modulators hybridized to silicon integrated circuits",*Applied Optics* **35** (8), pp. 1180-1186 (1996).

[Xu *et al.*, 1994] L. Xu, A. Kryzak, A. Yuille, "On Radial Basis Function Nets and Kernel Regression: Statistical Consistency, Convergence Rates, and Receptive Field Size", *Neural Networks* **7**, 609-628 (1994).

[Yayla *et al.*, 1994] G. Yayla, A. V. Krishnamoorthy, G. C. Marsden, S. C. Esener, "A prototype 3D optically interconnected neural network", *Proceedings of the IEEE* **82** (11), 1749-1762 (1994).

[Yang and Yu, 1992] X. Yang, F. T. Yu, "Optical implementation of the Hamming net", *Applied Optics* **31**, 3999-4003 (1992).

[Yang *et al.*, 1991] X. Yang, T. Lu, F. T. Yu, "Redundant-interconnection interpattern- association neural network", *Applied Optics* **30**, 5182-5187 (1991).

[Yang and Huang, 1994] G. Yang, T. S. Huang, "Human Face Detection in a Complex Background", *Pattern Recognition* **27** (1), 53-63 (1994).