

Expectation-Based Selective Attention

Shumeet Baluja
October 1, 1996
CMU-CS-96-182

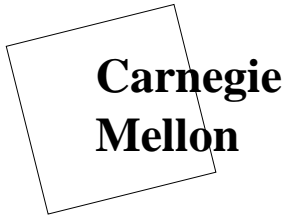
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.*

© by Shumeet Baluja

Shumeet Baluja was supported by a National Science Foundation Graduate Student Fellowship and a Graduate Student Fellowship from the National Aeronautics and Space Administration, administered by the Lyndon B. Johnson Space Center, Houston, TX. This research was also partially sponsored by the Office of Naval Research, under the Multiple University Research Initiative (MURI) contract #N00014-95-1-0591. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the National Science Foundation, National Aeronautics and Space Administration, Office of Naval Research, or the U.S. Government.

Keywords: Selective Attention, Vision, Expectation, Prediction, Relevance, Saliency Map, Neural Networks, Connectionism, Machine Learning, Autonomous Navigation, Mobile Robots, Hand Tracking, Semiconductor Wafer Fabrication, Face Detection, Object Detection.



School of Computer Science

DOCTORAL THESIS
in the field of
Computer Science

Expectation-Based Selective Attention

SHUMEET BALUJA

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

ACCEPTED:

MAJOR PROFESSOR

DATE

DEAN

DATE

APPROVED:

PROVOST

DATE

Dedicated to Sushil and Meenakshi

ABSTRACT

In many real-world tasks, the ability to focus attention on the relevant portions of the input is crucial for good performance. This work has shown that, for temporally coherent inputs, a computed expectation of the next time step's inputs provides a basis upon which to focus attention. Expectations are useful in tasks which arise in visual and non-visual domains, ranging from scene analysis to anomaly detection.

When temporally related inputs are available, an expectation of the next input's contents can be computed based upon the current inputs. A saliency map, which is based upon the computed expectation and the actual inputs, indicates which inputs will be important for performing the task in the next time step. For example, in many visual object tracking problems, the relevant features are predictable, while the distractions in the scene are either unpredictable or unrelated to the task. The task-specific selective attention methods can be used to create a saliency map which accentuates only the predictable inputs that are useful in solving the task. In a second use of expectation, anomaly detection, the unexpected features are important. Here, the role of expectation is reversed; it is used to emphasize the unpredicted features.

The performance of these methods is demonstrated in artificial neural network based systems on two real-world vision tasks: lane-marker tracking for autonomous vehicle control and driver monitoring, and hand tracking in cluttered scenes. For the hand-tracking task, techniques for incorporating *a priori* available domain knowledge are presented. These methods are also demonstrated in a non-vision based task: anomaly detection in the plasma etch step of semiconductor wafer fabrication.

In addition to explicitly creating a saliency map to indicate where a network should pay attention, techniques are developed to reveal a network's *implicit* saliency map. The implicit saliency map represents the portions of the input to which a network will pay attention in the absence of the explicit focusing mechanisms developed in this thesis. Methods to examine the features a network has encoded in its hidden layers are also presented. These techniques are applied to networks trained to perform face-detection in arbitrary visual scenes. The results clearly display the facial features the network determines to be the most important for face detection. These techniques address one of the largest criticisms of artificial neural networks — that it is difficult to understand what they encode.

ACKNOWLEDGEMENTS

It is a pleasure and an honor to acknowledge the role of my friends and family in the completion of this thesis. Without them, not only would I have not known how to finish this thesis, but I would not have known how to begin.

I am greatly indebted to my advisors and committee members, Dean Pomerleau, Tom Mitchell, Takeo Kanade and Tomaso Poggio. I was fortunate to have some of the people that I respect most guide me through my dissertation. They provided much more than excellent technical guidance. As anyone who has gone through the Ph.D. process knows, equally important to technical guidance is support, encouragement and friendship. I received much more of these than I could have expected. I thank all of them for taking me from a first year graduate student who felt in awe of their accomplishments, and felt that he had an impossible amount to learn from each of them, to a Ph.D. who feels exactly the same way.

During the four years that I was a graduate student, I spent many nights with friends doing everything from discussing Kalman Filters to playing Doom. Because some of them are students who have not yet finished their thesis, and their advisors might read this, I will not mention with which of them I discussed technical issues and with which I played games. Let me, instead, just say that it was a pleasure spending time with Justin Boyan, Tammy & Ranes Carter, Rich Caruana, Mei Chen, Rob DeLine, John Hancock, Jennie Kay, David LaRose, Jim Rehg, Jeff Schneider, Michael Smith, and Rahul & Gita Sukthankar. In particular, I would like to thank Henry Rowley, who has been a tremendous amount of fun to work and play with, and has provided countless hours of invaluable discussion on the topics presented here. This thesis has greatly benefited from his help.

On a more personal note, I would like to thank Kaari Flagstad for her unwavering and unconditional friendship through these years. She is undoubtedly one of the most patient and kind people that I have ever met. Although we were both graduate students, she gave unselfishly of her time. She gave me the freedom to get immersed in my work when I needed to, and yet was able to keep me in contact with the rest of the world when I started getting lost. I hope that I will be able to someday repay her for all that she has done for me.

Finally, I would like to thank my parents. In every sense, without them, none of this would have been possible. In the 25 years that they have put up with me, they have shown me nothing but love and encouragement. It is because of the excitement for knowledge that they gave to me when I was a child that I wanted and needed to get a Ph.D. It is impossible to say all that should be said here; let me conclude by saying that I cannot begin to thank them enough, and that they will always have my respect and love.

Shumeet Baluja
October 1, 1996

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION	1
Motivation and Background	2
Psychophysical & Computational Models of Selective Attention	9
Overview of Thesis	12
 CHAPTER 2 : TASK-SPECIFIC FOCUS OF ATTENTION	 17
Introduction	18
Using the Network's Hidden Layer to Determine Task-Specific Importance	19
<i>Using the Differences Between Expectation and Realization</i>	23
Using Expectation to Remove Irrelevant or Distracting Features	25
<i>Relations to Other Recurrent Neural Networks</i>	28
Empirical Results: Lane Tracking	29
<i>The ALVINN Road Following System</i>	30
<i>Experiment Details</i>	31
Other Network Architectures for Focusing Attention	38
Summary	39
 CHAPTER 3: UNDERLYING ASSUMPTIONS FOR TASK-SPECIFIC EXPECTATION-BASED SELECTIVE ATTENTION	 41
Overview	42
<i>The Experiments</i>	44
Discontinuous Cross Detection	44
<i>Experiment 1: Random Noise</i>	45
<i>Cross Detection with Temporally Coherent Noise</i>	48
Encoding State – Not Position	50
Combined Detection and Recognition Tasks	57
<i>Two Supervised Signals: Location and Identification</i>	60
<i>One Supervised Signal: Identification</i>	62
<i>One Supervised Signal: Location</i>	67
<i>Ghost Features Due to Incorrect Predictions</i>	70
<i>Summary of Combined Detection and Recognition Tasks</i>	71
Why Not Use Principal Components Analysis?	72
<i>Discussion</i>	75
<i>Ramifications for Anomaly Detection</i>	76
Where's the Catch?	78
Chapter Summary	79
<i>Future Experiments</i>	80

CHAPTER 4: UTILIZING DOMAIN-SPECIFIC KNOWLEDGE FOR CREATING EXPECTATIONS	83
Introduction	84
Vision-Based Hand Tracking	85
Incorporating Domain Specific Knowledge	88
<i>Explicitly Creating Expectations</i>	88
<i>Post-Processing the Output</i>	93
Results	94
Conclusions and Future Experiments	98
 CHAPTER 5: REVERSING THE ROLE OF EXPECTATION: ANOMALY DETECTION	 101
Reversing the Role of the Expected	102
Introduction to the Problem	103
Experiment Specifics	104
<i>The Plasma Etcher</i>	104
<i>Data</i>	104
<i>Faults to Be Detected</i>	105
<i>Artificial Neural Network Training & Testing Specifics</i>	106
Methods for Fault Detection & Classification	109
<i>Method A: Base-Line</i>	109
<i>Method B: Using the Previous Wafer for Comparison -</i>	
<i>Incorporating Machine-State Information</i>	111
<i>Method C: Using State & Individual Networks</i>	113
<i>Method D: Computing Expectations</i>	114
<i>Method E: Computing Expectations and Individual Networks</i>	119
<i>Summary of Empirical Results</i>	120
Why Using Expectation Improves Performance	123
Conclusions & Future Directions	125
 CHAPTER 6: EXPECTATIONS WITH NON-TEMPORAL DATA	 129
Introduction	130
Face Detection	131
Encouraging Distributed Input Reliance	135
Analysis of Networks Trained for Face Detection	137
<i>Networks Trained Without Noise</i>	137
<i>Networks Trained with "Block-Noise"</i>	138
<i>Networks Trained with Uniform Noise</i>	141
Summary and Conclusions	143

CHAPTER 7: RELATED WORK	147
Alternate Neural Approaches for Selective Attention	148
Task Transfer	151
<i>Multitask Learning</i>	153
<i>Learning from Other Learners</i>	154
Link-Predictive Neural Networks	155
Integrating Saliency Over Time - Input Relevance	156
Kalman Filters	160
Active Vision	162
Dickmanns' 4D Approach to Dynamic Vision	164
An Object-Based Approach	166
 CHAPTER 8: CONCLUSIONS AND FUTURE DIRECTIONS	 169
Summary & Contributions	170
<i>Creating Expectations and Expectation-Based Saliency Maps</i>	170
<i>Common Features of Successful Applications</i>	172
<i>Methods for Interacting with a Neural Network</i>	173
<i>Visualizing the Contents of a Neural Network</i>	174
<i>The Role of Neural Networks for Expectation-Based Selective Attention</i>	175
Future Work	176
<i>Pixel-Based Filtering</i>	176
<i>Parallel vs. Serial Search</i>	177
<i>Stability in Training</i>	178
<i>Incorporating More State Information</i>	179
<i>Experiments with Human Expectation-Based Selective Attention</i>	179
<i>Future Application Areas</i>	182

CHAPTER 1

INTRODUCTION

The thesis of this research is that, for temporally related inputs, an expectation of the next time step's inputs can provide a basis upon which to focus attention in scene analysis and anomaly detection. This is most easily understood when examined in the context of human behavior. When performing actions in the real world, humans create expectations of what the world will be like in the proceeding few moments, and beyond. The presence, positions, and orientations of objects and their attributes are often predictable for long periods of time. Therefore, having analyzed the input scene once, we can determine where to focus our attention in the future. We are free to concentrate on only the portions of the scene which are interesting. We can define "interesting" as we wish, including the portions of the scene which are necessary for taking us towards our goals, or portions of the scenes which we did not expect to see.

1. MOTIVATION AND BACKGROUND

Many real world tasks have the property that only a small fraction of the available input is important at any particular time. On some tasks this extra input can easily be ignored; however, often the similarity between the important input features and the irrelevant features is great enough to interfere with task performance. Two common examples of this phenomena are speech recognition in a noisy environment and image processing of a cluttered scene; for example, see Figure 1-1. In both cases, the extraneous information in the input can be easily confused with the important features, making the specific task much more difficult.

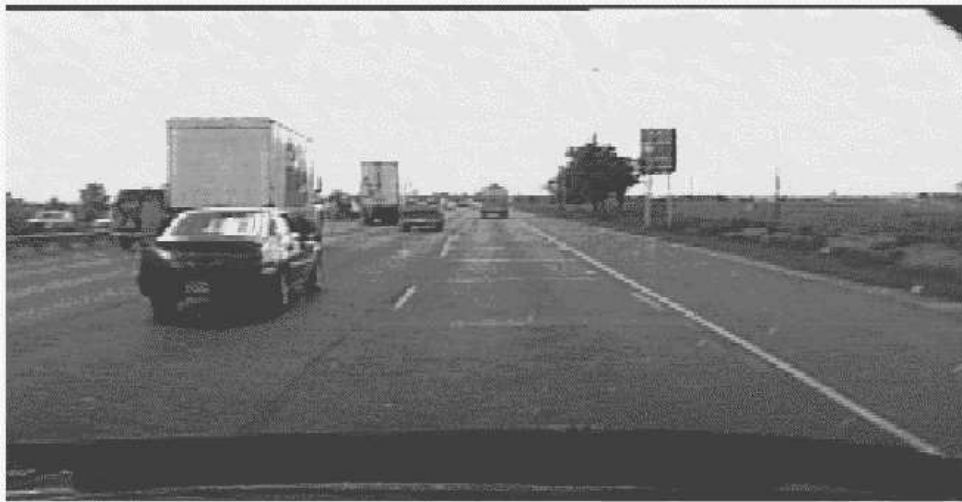


Figure 1-1: Typical scene to be processed in vision-based autonomous navigation tasks.

Although the extra inputs may not directly affect the relevant portions of the input signal, processing the signal may nevertheless become more difficult since the inputs become more complex. For example, consider finding or tracking a moving object against an empty background in comparison to a changing, cluttered one. The values in the irrelevant inputs can make it difficult to determine which inputs are relevant and which are not. One of the methods by which humans function in the presence of many distracting features is to selectively attend to only portions of the signal. A commonly accepted theory is that the limited processing capacity of humans is a fundamental constraint which

imposes the need for selectivity in processing [Allport, 1989][Broadbent, 1982]. Although a machine may not be subject to the same limitations as a human, distracting features remain harmful. Most machine learning, artificial intelligence, and vision based algorithms are not robust to unanticipated forms of noise, and performance improves with fewer distracting features in the input.

To discriminate between relevant and irrelevant inputs, it is first necessary to understand the requirements of the task. Importance of particular features, or “saliency” as defined in this work, is task-specific. For different tasks, different subsets of the same inputs may be important. A notion of relevance must be able to account for the task-specific requirements of the particular problem. It should be noted that, in general, saliency does not necessarily entail task-specificity. Although not pursued here, saliency can also be discussed in the context of a learning algorithm rather than a task. For a learning algorithm, a training example may be irrelevant because it contains no more information for that algorithm than was present in the training set without the example. As a second example, certain inputs may contain only noise, thereby giving no information to learning algorithms, regardless of the task.

Many studies of input relevance make the assumption that the best selected features for each decision point is the minimal set of features which is needed to classify every point [Caruana & Freitag, 1994][John, *et al.* 1994][Langley & Sage, 1994]. In those studies, relevancy of the inputs is determined on the scale of the training/validation sets, not individual examples. However, in many domains, such as vision, the smallest set of inputs needed to classify every example may be the entire set of inputs, while the set needed to classify each individual example may be very small. For example, again consider visually tracking a moving object across a large pixel-based input. The goal may be to report the value of some attribute of the object, such as orientation. The inputs which are relevant to the task will change as the object moves. Assuming the object appears everywhere in the input with equal probability, no input can be considered more important than another when the entire training set is considered. However, when individual examples are considered, the position of the object gives relevance information; only the pixels which correspond to the position of the object are relevant. In many vision applications, choosing

the smallest subset of relevant points at the training set level will provide little or no filtering. Assessment of input relevance for many vision tasks must be done dynamically rather than as a pre-processing step.

The studies in this thesis are most easily classified in the vision literature as either focusing attention, intelligent filtering, or saliency detection. In the artificial intelligence literature, this work has relations to the study of *relevance*. Selecting portions of the scene is a form of allocating relevance to particular inputs at different times. Relevance information can be incorporated into machine learning or biological systems by pre-attentively or attentively selecting the regions of the input on which to focus.

In this work, “relevant” or “salient” is defined operatively: an input is considered relevant to a task when it improves task performance. This definition makes “relevant” dependent upon the inductive algorithm used to solve the task. As mentioned before, many machine learning algorithms improve generalization when only the relevant features are given to them, such as ID3/C4.5 [Quinlan, 1993] [Caruana & Freitag, 1994]. An input can be considered irrelevant when it either has no effect, or degrades, performance of the learning algorithm. There are at least two reasons that irrelevant inputs can degrade performance. First, extra inputs increase the dimensionality of the input space. This can make learning, for the purpose of good generalization, more difficult if there are only a limited number of training examples. Second, the irrelevant inputs can contain distracting or misleading information. It is this second problem which will be of primary concern in this study.

In the domains explored in this study, it is necessary to dynamically re-evaluate the amount to focus on each input. Therefore, it is not sufficient to simply pre-process the inputs by static pruning methods [Alumualim & Dietterich, 1991][Caruana & Freitag, 1994][John *et al.*, 1994]. The attention of the processing system is dynamically focused only on the relevant inputs by masking out noise or distracting features. Because of the potentially high degree of similarity between the relevant features and the noise, this filtering is often extremely difficult. A means by which humans perform well, even in the presence of large amounts of noise, is through the use of *expectations* to guide where to focus attention, or how to filter the input. Once the important features in the current

input are found, an expectation of what the important features in the next inputs will be, as well as where they will be, can be formed. Dickmanns summarizes the use of expectations for scene processing as follows:

... A physical object cannot be at two different locations at the same time. In order to move from one location to another, energy is required for acceleration and deceleration, and time is required because the energy available to effect locomotion is bounded. These facts constitute constraints on the motion process which may help considerably when tracking the locomotion of objects, especially when accelerations and decelerations are very limited in magnitude as is the case in most of the occurrences in our natural and even technical environments. ..

Therefore, if we have a good internal representation of a situation in our environment we are in a much better position to understand the next image of a real-time sequence provided this representation allows us to predict how the process under observation is going to evolve over time taking certain control or perturbation inputs into account. If this prediction model is approximately correct one can concentrate the limited data processing capabilities on the data originating in the local environment of the predicted spot, thereby making the sensing process much more efficient; in addition, the data processing algorithms may also be adjusted to the predicted situation, thereby further increasing efficiency. This positive feedback favors the evolution of powerful prediction capabilities since, in spite of additional computing resources required for prediction, the overall requirements may be decreased for the same performance level; on the other hand, completely new performance levels and a deeper understanding of environmental processes may be achievable with this approach. [Dickmanns, 1992, page 303-304].

This notion of importance is task-specific; different tasks can require the processing of different features in the same input. This knowledge reduces the difficulty of the prediction task. *Expectations do not need to be formed for all of the features*; rather, expectations are only required for the relevant inputs. For example, consider driving in a car. If you are the driver, the regions of your visual input to which you pay attention will be largely focused on the road; you do not need to attend to the buildings on the side of the road, etc. On the other hand, if you are a passenger looking for a restaurant or reading road signs, you may be more likely to pay attention to billboards, signs, or buildings, etc., rather than the road ahead, see Figure 1-1. The task determines which inputs are important.

How do people know what to pay attention to? Let's again examine the task of driving from the viewpoint of the driver. Given an image of the road ahead, the locations of the important features for staying on the road can usually be reliably predicted for the next time-step. These expectations can be used in two manners. In one use of expectation, any portions of the input which are inconsistent with these expectations can be de-emphasized from processing. This de-emphasizes unexpected features. By removing unexpected features, the effects of distractions in the periphery and on the road ahead are reduced, while the road's features become more prominent. Therefore, the task of staying on the road becomes easier. Of course, the potential problem in using expectation to remove unexpected features is that many surprise features, such as other cars or objects on the road, should not be removed from the input. Although these objects may not be important for staying in your lane, they are important when considering the full task of navigation. These objects will need to be processed in order to avoid collisions. To address these needs, expectation can also be used, in parallel, in an opposite manner. It can be used to emphasize anything which does *not match* expectation. This second use of expectation emphasizes unexpected features. A high level schematic of the use of expectation is shown in Figure 1-2. Note that if we allow memory or state information to be preserved in the system, the expectation can be formed based upon more than the single previous input.

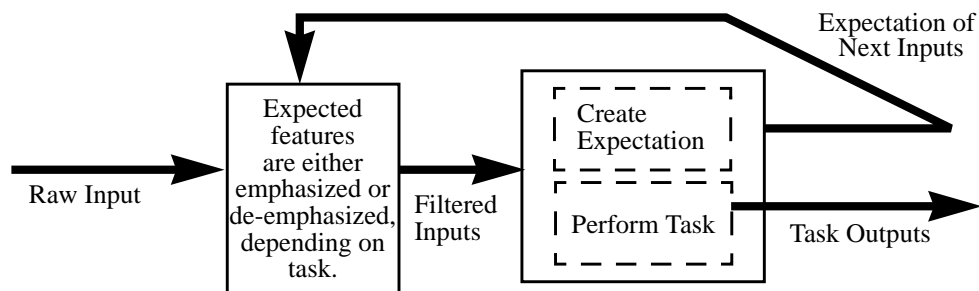


Figure 1-2: High-level schematic of the use of expectation. Note that the process which creates the next time step's expectation and the process which solves the task are closely related since the expectation which is created is task-specific. The module to create expectation and perform the task may contain memory or state information.

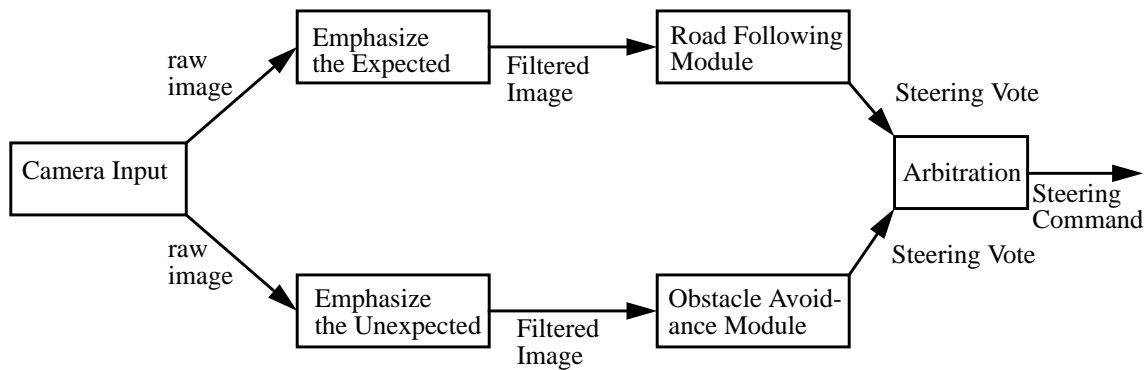


Figure 1-3: A model for using both forms of expectation in the driving domain.

The motivation for employing this bipolar usage of expectation is largely pragmatic; it maps well to the structure of many real-world machine learning and adaptive systems. For example, consider Carnegie Mellon’s Navlab autonomous navigation system. The road-following module (ALVINN, [Pomerleau, 1993]) is separate from the obstacle avoidance modules [Thorpe, 1991]. Each module can use different inputs, or different input filtering algorithms, to match its specific requirements. The first role of expectation, in which unexpected features are de-emphasized, is appropriate for the road-following module. The second role of expectation, in which unexpected features are emphasized, is appropriate for the obstacle avoidance modules. A higher level system arbitrates between these and other modules. See Figure 1-3.

A second example which demonstrates the applicability of the bipolar uses of expectation is a system which locates and tracks a person in a room. Initially, when nobody is in the room, anything unexpected which appears in the visual field should be attended, since it might be a person who should be tracked. If the unexpected object is determined to be a person, expectations of the person’s location, which are based on estimated position and velocity, can be used to track the person’s movements.

There are other problems which require only a single use of expectation. For example, consider anomaly detection. In this domain, it is the portions of the input which do not match expectation that should be emphasized for processing, since these have a higher

potential for being anomalies. This interpretation of expectation will be explored in Chapter 5, in which task-specific predictions were used to detect and classify anomalies in the plasma etch step of semiconductor wafer fabrication.

In deciding whether the approaches described above are suitable to a new problem, there are three main criteria which must be considered:

1. Consecutive inputs in time must have temporal coherence.
 - The first use of expectation is to remove distractions from the inputs. Given the set of current inputs, the activations of the relevant inputs must be predictable while the irrelevant inputs are either unrelated to the task or are unpredictable. As shown in Chapters 2-4, in many visual object tracking problems, the relevant inputs are often predictable while the distractions are not. Even in the cases in which the distractions are predictable, as long as they are unrelated to the main task, these methods can work.
 - The second use of expectation is to emphasize unexpected or potentially anomalous features. Given the set of current inputs, the activations of the relevant inputs should be unpredictable while the irrelevant ones are predictable. As described in Chapter 5, this is often the case for anomaly detection tasks.

This dual use of expectations will be expanded upon in much greater detail throughout this thesis. Although some of the developed relevance analysis tools are applicable to non-temporal data, as shown in Chapter 6, the majority of this thesis will address temporal data.

2. When expectations are used as a filter, it is necessary to explicitly define the role of the expected features. In particular, it is necessary to define whether the expected features should be considered relevant or irrelevant, and therefore, whether they should be emphasized or de-emphasized, respectively. As described previously, the role of expectation can change depending upon the task and the state of the system.
3. There should be a method of de-emphasizing features. In many vision systems, features can be de-emphasized by reducing their activation to a “background” intensity or to their average value over a large number of examples. Even if there is no explicit method of de-emphasizing features, the predictions themselves can be used as extra inputs to a learning system. This will be explored in Chapter 5.

After describing the applications and results, we will return to these criteria to see how

the applications explored in this thesis fit with these criteria (Chapter 8).

Although biological plausibility was not a primary goal in this thesis, psychological studies which have explored attention in humans provide a good background for this material. A brief introduction to some of the psychophysical models and issues related to selective attention is given in the next section.

2. PSYCHOPHYSICAL & COMPUTATIONAL MODELS OF SELECTIVE ATTENTION

Focus of attention has been studied in a variety of contexts. One of the largest branches of study has examined attention in static images. For example, [Triesman and Gelade, 1980] describe a study in which a subject recognizes the letter ‘**S**’ mixed in a field of ‘**X**’s and ‘**T**’s of various colors. The ‘**S**’ *pops out* to the subject, suggesting a pre-attentive and parallel processing of the image. Pop-Out happens when there is a single distinctive feature of the target; in the case of the ‘**S**’, the feature is its curvature. If an object must be distinguished by “conjunctive” features such as color and shape, the search seems to be performed in a more serial fashion [Triesman and Gelade, 1980][Hurlbert and Poggio, 1985]¹. For example, consider finding a green ‘**T**’ among a field of green ‘**X**’s and brown ‘**T**’s. The most commonly accepted analogy for this search process is termed the “spotlight” hypothesis. The spotlight moves from one location to another; the areas in which it focuses are operatively defined to be the areas in which an improved performance is found in the tasks of stimulus detection, identification, localization, or simple and choice response times [Umla, 1988].

Why is attention needed? The need for selective attention has been attributed to limiting processing capabilities [Broadbent, 1982]. The spotlight, or filter theory, was used to explain how information is excluded from further analysis. This theory was later slightly modified to change exclusion of sensory inputs to attenuated sensory inputs. Nonethe-

1. More recently, Triesman’s feature integration theory has been modified to show that certain conjunctions (e.g. motion and stereoscopic disparity) can be detected in speeds more indicative of parallel search, and that the attentional spotlight does not necessarily only work in the spatial dimension, see [Triesman, 1988]. For example, if one is searching for red objects, all non-red objects may be filtered out, so that a conjunction search would occur only on the red objects [Olshausen & Koch, 1996].

less, the ideas of limited capacity remained unchanged [Allport, 1989]. Alternate views of serial processing have also been suggested. Ullman presents an argument that serial processing is essential, since it is “imposed by the inherent sequential nature of various visual tasks”, such as “inside-outside” relations [Ullman, 1980].

The question of where this filtering takes place has been studied in depth (see [Broadbent, 1971][Deutsch & Deutsch, 1963][Allport, 1989][Hoffman, 1986] – to name a few). The controversy lies in whether there is *early* or *late* selection. Broadbent has suggested that it takes place early, in very low level processing. These theories suggest that selection is made on the basis of criteria like spatial location, rather than object identification. In late selection theories, the selection of stimuli occur after a process analogous to pattern recognition has been applied unselectively to all of the input; these theories assume that there are no capacity limitations and that all sensory messages which impinge on the organism are analyzed at the highest level [Deutsch & Deutsch, 1963]. In this theory a message will reach the same perceptual and classifying mechanisms whether attention is paid to it or not [Van Der Heijden, 1992]. For an overview of these topics, see [Pashler & Badgio, 1985] and [Van Der Heijden, 1992].

Computational models of the spotlight mechanism have been proposed in the context of artificial neural networks [Koch and Ullman, 1985][Mozer, 1988], to name a few. Mozer makes the distinction between “data-driven” and “conceptually-driven” guidance of the spotlight. A simple case of “Data-Driven” guidance is that the spotlight should be drawn to objects, but not to empty spaces in the visual field. A “Conceptually-Driven” spotlight is controlled directly by a “higher level of cognition” (goal driven behavior). For example, this is necessary when reading, in which text must be scanned from left to right.¹

Many of the studies described above have concentrated on attention in static images. In sequences of images, the process of focusing attention becomes more challenging; objects can move and change. Nonetheless, people can routinely solve the problem of focusing, and maintaining, attention on moving and changing objects. The ability to do this with

1. It should be noted that moving the “spotlight of attention” does not necessarily entail eye movements. Umiltà and Posner discusses the relationship between eye movements and focus of attention [Umiltà, 1988][Posner *et al.*, 1980]. The speed at which the attentional spotlight can move from one location to another has been hypothesized to be as fast as four to six times faster than eye movements, see [Olshausen & Koch, 1996] for a summary of several viewpoints.

“odd-man-out”¹ features has been called *indexing* [Ullman, 1980]. The ability to index is a prerequisite for visual motor coordination and object description [Trick & Pylyshyn, 1991]. Allport describes a form of selection based on action. Since humans can generally direct action to only one of many simultaneously available objects, this form of selection must decide on where to direct the action:

Any goal-directed action requires the specification of a unique set of (time-varying) parameters for its execution - parameters that determine the outcome as *this* particular action rather than any other...

Suppose that visual information has to guide manual reaching, for example to grasp a stationary object or to catch a moving one. Clearly, many different objects may be present in the visual field, yet information specific to just *one* of these objects must uniquely determine the spatiotemporal coordinates of the end-point of the reach, the orientation and opening of the hand, and so on. Information about the positions, sizes, and the like of the other objects within view and also available, must not be allowed to interfere with (that is, to produce crosstalk affecting) *these* parameters - though they may need to influence the trajectory of the reach in other ways. Consequently some *selective* process is necessary to map just those aspects of the visual array, specific to the target object, selectively onto appropriate control parameters of the action. I have termed this the functional requirement of *selection-for-action*... [Allport, 1989, page 648].

“Selection-for-action” introduces the task/action-specific nature of focusing. Information about irrelevant features or objects must be filtered out, to avoid crosstalk and confusion with respect to the feature or objects of interest. This model relates closely to the model presented in this thesis.

The goal for the focus of attention mechanism used in this thesis was to create a conceptually driven *expectation* which can maintain attention on moving objects which may change shape, orientation and position. The expectation can be used to filter the *next set of inputs*. Since different tasks will require analyzing different portions of the inputs, the expectation, and therefore the focus of attention, must be task-specific (selection-for-action). The focus of attention must designate as relevant only the portions of the input which are necessary for completing the task.

1. These are features which are distinguished by attributes such as color, or by motion in an otherwise stationary background.

The model presented in this work is “conceptually” driven. It combines aspects of early and late selection. Late-selection is necessary since the decision of which inputs to filter in the next time step is based upon the specific task and the objects/features detected in the current input. Therefore, initially, the inputs must all be processed. However, once the expectation of the next inputs is computed, it can then be used in an early-selection manner. For example, if a person is tracking a moving object, such as an automobile, knowledge of the car’s movements and the initial placement of the car in the visual scene give information of which inputs will be relevant next. Therefore, the next inputs can be filtered at a very low level based upon their position in the input retina.

It should be noted that the traditional spotlight analogy will not hold in many of the tasks examined. For example, if a task requires two portions of the input scene to be analyzed concurrently, multiple regions can be focused upon concurrently. In the model presented in this paper, the relevant portions of the scene do not have to be spatially continuous over time. Therefore, unlike many traditional theories of spotlight-based focus of attention, the filter may encompass multiple objects and multiple disjoint portions of the scene either at the same time or in consecutive time steps; see [Posner *et al.*, 1980][Umla, 1988].

3. OVERVIEW OF THESIS

The goal of this thesis was to invent and investigate new techniques for creating expectations of future inputs. Once these techniques were developed, by examining the requirements of real-world tasks, methods were developed to use these expectations for intelligent, adaptive, filtering. Several methods are presented. They differ in the amount of domain-specific knowledge which can be integrated and the manner in which expectations provide information about where to pay attention. In this thesis, four real-world tasks are examined. Each of these tasks accentuates different aspects of the focus of attention methods, and will be described in detail. The thesis is organized as follows:

- **CHAPTER 2: TASK-SPECIFIC FOCUS OF ATTENTION**

This section describes the use of task-specific expectations to determine the salient portions of the next set of inputs. The notion of a saliency map is introduced. The saliency map is used to eliminate noise, or distracting features, from the input. This method is applied to the task of vision-based lane tracking for autonomous vehicle control or vehicle monitoring, using the ALVINN system [Pomerleau, 1993].

- **CHAPTER 3: UNDERLYING ASSUMPTIONS FOR TASK-SPECIFIC EXPECTATION-BASED SELECTIVE ATTENTION**

In this chapter, some of the underlying assumptions of the task-specific expectation-based selective attention methods are examined. The focus of attention methods rely on the ability to predict the next input scene. This chapter explores the requirements for making accurate predictions. This chapter explains, through a series of carefully constructed synthetic examples, the type of information which the hidden layer must encode for the prediction mechanisms to perform correctly.

- **CHAPTER 4: UTILIZING DOMAIN-SPECIFIC KNOWLEDGE FOR CREATING EXPECTATIONS**

The saliency map is presented as a method of directing the network's processing. In Chapter 2, the saliency map was automatically derived. However, if *a priori* information of where the important features are located in the input is available, this information can be directly used to explicitly modify the saliency map. The saliency map is presented as a general mechanism for integrating symbolic rules which indicate the importance of inputs with neural processing.

- **CHAPTER 5: REVERSING THE ROLE OF EXPECTATION: ANOMALY DETECTION**

This chapter describes the use of expectation to emphasize anomalies in sequential data. This method is applied to the detection of anomalies in the plasma-etch step of semiconductor wafer fabrication. This task demonstrates the ability of the architectures and algorithms developed in this thesis to work outside of visual domains. This chapter also presents methods to use expectations which are not based on explicit filtering.

- **CHAPTER 6: EXPECTATIONS WITH NON-TEMPORAL DATA**

Techniques are developed to reveal a network's *implicit* saliency map. The implicit saliency map represents the portions of the input to which a network will pay attention in the absence of the explicit focusing mechanisms developed in this thesis. Methods to examine the features a network has encoded in its hidden layers are also presented. These techniques are applied to networks trained to perform face-detection in arbitrary visual scenes. The results clearly display the facial features the network determines to be the most important for face detection.

- **CHAPTER 7: RELATED WORK**

Prediction-based selective attention is related to a variety of other work both inside and outside of the neural network field. Another model of network learning, termed *multitask learning*, is explained, and the relations to it are drawn. This work has close relations to the work done in the field of *relevance*. Active vision, Kalman Filters, and more traditional methods of vision-based focus of attention are also discussed.

- **CHAPTER 8: CONCLUSIONS, CONTRIBUTIONS & FUTURE DIRECTIONS**

This chapter presents a review and comparison of the different techniques explored throughout this thesis. Also presented are several directions for future research, including new applications for expectation-based selective attention.

CHAPTER 2

TASK-SPECIFIC FOCUS OF ATTENTION

In many real-world tasks, the ability to focus attention on the important features of the input is crucial for good performance. In this chapter, a mechanism for achieving task-specific focus of attention is presented. A saliency map, which is based upon a computed expectation of the contents of the inputs at the next time step, indicates which regions of the input retina are important for performing the task. The saliency map can be used to accentuate the features which are important and de-emphasize those which are not. The performance of this method is demonstrated on the real-world robotics task of vision-based lane tracking.

1. INTRODUCTION

One of the main contributions of this thesis is to use the representation of a neural network's hidden layer, trained to perform a time sequential task, to make task-specific predictions of what the next inputs will be. These computed expectations of the next input can be used to provide a mechanism for focusing the network's attention on important features. Throughout the rest of this chapter, focus of attention will be discussed in the context of artificial neural networks (ANNs). Nonetheless, it should be understood that the use of ANNs is an implementation decision. Other methods can be used if they provide a task-specific reduction of the inputs from which expectation of the next inputs can be derived. For example, statistical regression methods reveal which inputs are important for solving particular tasks; therefore, these inputs can be used create task-specific expectations. As explored in Chapter 4, in the task of vision-based hand tracking, symbolic rules can also be used to create expectations.

In order to direct processing to only the relevant portions of the input, the relevant portions must first be determined. Saliency maps are tools used to indicate the relative importance of different regions of the input scene. In many studies, saliency maps have been constructed in a bottom-up manner [Clark & Ferrier, 1992][Koch & Ullman, 1985]. In its simplest form, the saliency map may contain only a binary value for each input, which indicates whether the particular input is relevant. More commonly, a real valued weight, signifying relevance, is associated with each input. A very cursory summary of the bottom-up approach is that multiple different task-specific feature detectors are placed around the input image. Each type of feature detector may contain a weight associated with it, to signify the relative importance of the particular feature. The regions of the image which contain high weighted sums of the detected features are the portion of the scene which are focused upon. Top-down knowledge can be incorporated in deciding which features are used, the weightings of the features, and how many regions are focused upon [Ahmad, 1991]. Another method of creating a saliency map is to emphasize all portions of the input which are different from their surrounding inputs; this was explored in [Koch & Ullman, 1985].

Although saliency maps are integral to the techniques presented in this study, the

approach used to create and use them is very different than the one described above. The features and their weightings are developed as the neural network learns to use the features. No *a priori* top-down information is assumed. In the method proposed here, the expectation of what the features will be in the *next* frame plays a key role in determining which portions of the visual scene *will be* focused upon.

Simultaneously learning to perform a vision-based task and filtering out clutter in the input scene is doubly difficult because of the “chicken-and-egg problem.” It is hard to learn which features to attend to before knowing how to perform the task, and it is hard to learn how to perform the task before knowing on which features to attend. This chapter describes a technique to solve this problem. It involves deriving a “saliency map” of the image from a neural network’s internal representation, as it is being formed, which highlights regions of the scene considered to be important. The saliency map, which is a function of the current input image and the previous input images, is used as feedback to focus the attention of the network’s processing on subsequent images. The proposed technique overcomes the chicken-and-egg problem by simultaneously learning how to identify which aspects of the scene are important, and how to use them to perform a task.

2. USING THE NETWORK’S HIDDEN LAYER TO DETERMINE TASK-SPECIFIC IMPORTANCE

The first step in creating a conceptually based saliency map (one which is driven by the task to be performed) is determining which portions of an input image are important. The method described in this section is based on *Input Reconstruction Reliability Estimation (IRRE)*, proposed by [Pomerleau, 1993]. IRRE is used to estimate how reliable a network’s outputs are. The reliability estimation in IRRE is made by using the hidden units to both reconstruct the input image and complete the main task. The heuristic used to relate the reconstructed inputs to the task output is the greater the similarity between the actual input image and the reconstructed input image, the more the internal representation has captured the input features, and therefore the more reliable the network’s response. This has been shown to work well empirically [Pomerleau, 1994]. Figure 2-1 provides a schematic of IRRE. Note that the weights between the input and hidden layers are trained to

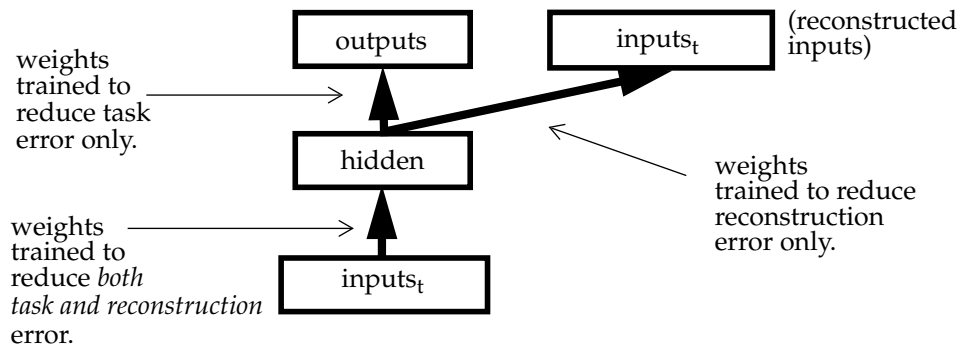


Figure 2-1: IRRE: using the hidden layer to encode information for reconstruction and task completion.

reduce *both* task and reconstruction error.

IRRE has successfully been used as a method of arbitrating between multiple “expert” networks. For example, consider the case in which one network (Network-1) was trained to steer an autonomous vehicle on a single lane road, and another on a double lane road (Network-2). The reconstruction of the inputs on a two lane road will be very poor for Network-1, while Network-2 will be able to accurately reconstruct the inputs. The performance will switch on single lane roads. The reconstruction error provides a method to determine which network to trust when road-types change. As pointed out by Pomerleau, a potentially large drawback of IRRE is the use of the hidden layer to encode all of the features in the image, rather than only the ones required for solving the particular task [Pomerleau, 1994]. This problem is addressed below.

The creation of a saliency map is based upon a very basic analysis of the neural network. The underlying premise is that if a strictly layered (connections are only between adjacent layers) feed-forward neural network can solve a given task, the activations of the hidden layer contain, in some form, the important information for this task from the input layer. One method of finding out what information is contained within the hidden layer is to attempt to reconstruct the original input image, *based solely upon the representation developed in the hidden layer*. Like IRRE, the input image is reconstructed from the activations of the units in the hidden layer. *Unlike IRRE, the hidden units are not trained to reduce reconstruction error, they are only trained to solve the particular task*. The input recon-

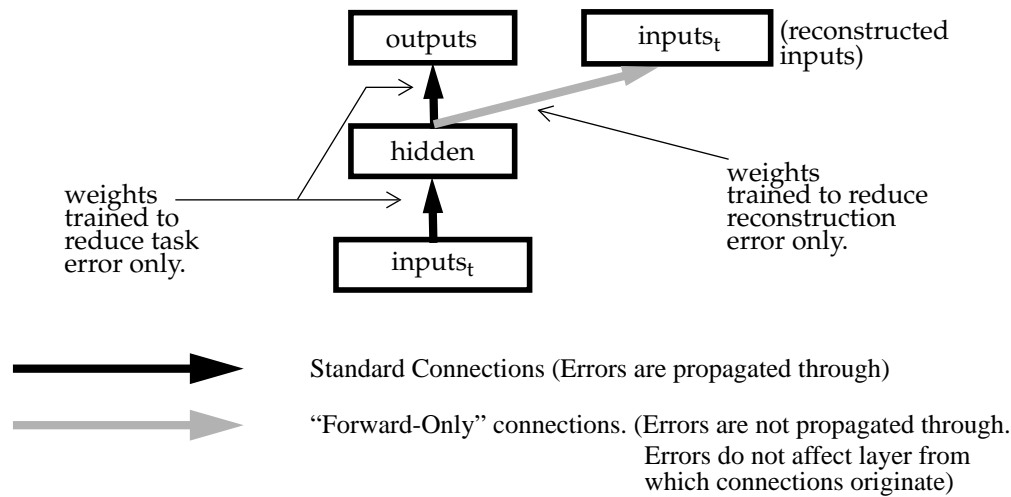
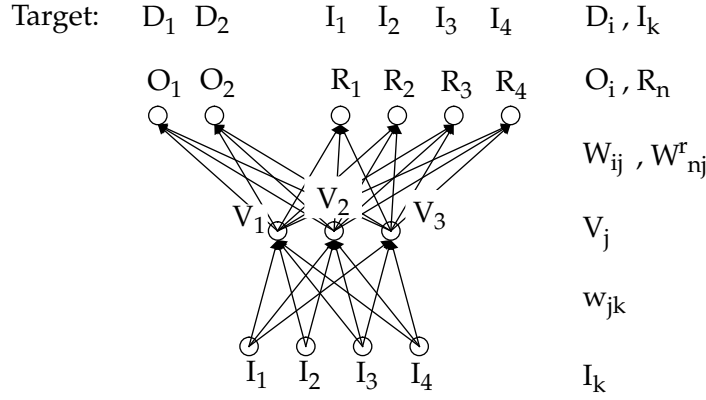


Figure 2-2: Using the activation of the hidden layer to reconstruct the input. The weights between the input and hidden layers are trained to only reduce task error, *not* reconstruction error. Extra hidden layers for reconstruction can be used.

struction is done by changing the weights from the network’s hidden layer to the reconstruction outputs only. The weights from the inputs to the hidden layer are *not* affected by the errors in reconstruction. Therefore, all of the capacity and representations developed in the hidden layer is devoted only to solving the task. See Figure 2-2.

The inputs will not be perfectly reconstructed in the output layer because the weights of the network between the input and hidden layers are not trained to perform reconstruction; they are trained to perform the task. The weight update rules are shown in Figure 2-3. The information which is encoded in the activations of the hidden units corresponds to the information which the network uses to solve the task. *Information which is not relevant to the task will not be encoded in the hidden units.* Since the reconstruction of the inputs is based solely on the hidden units’ activations, and the irrelevant portions of the input are not encoded in the hidden units’ activations, the irrelevant portions of the input *cannot* be reconstructed. One exception to this is that features which are always present, such as inputs which contain nearly constant values over time, can be reconstructed, regardless of relevance. However, these can easily be eliminated through other procedures, such as subtraction of the average image. Additionally, it should be noted that



O_i :Task Output (after transfer function)

D_i :Desired Task Output

R_n :Reconstruction Output ($n = k$)

I_k :Input to the Network

$g(x)$:Transfer Function

V_j :Output of the Hidden Units (after transfer function)

W_{ij} :Weights between hidden unit j and output i

W_{nj}^r :Weights between hidden unit j and reconstruction output i

w_{jk} :Weights between input k and hidden unit j

h_j :summed input to the hidden unit $j = (\sum_k w_{jk} I_k)$

h_i :summed input to the output unit $i = (\sum_j W_{ij} g(h_j))$

h_n :summed input to the reconstruction unit $n = (\sum_j W_{nj}^r g(h_j))$

η :Neural Network Learning Rate

The connections between the Task Output Units and the Hidden Units are trained as follows (shown for one output):

$$\Delta W_{ij} = -\eta \frac{\partial E}{\partial W_{ij}} = \eta [D_i - O_i] g'(h_i) V_j$$

The connections between the Reconstruction and the Hidden Units are trained as follows (shown for one output):

$$\Delta W_{nj}^r = -\eta \frac{\partial E}{\partial W_{nj}^r} = \eta [I_n - R_n] g'(h_n) V_j$$

The connections between the Hidden Units and the Input Units are trained as follows (shown for one hidden unit):

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} = -\eta \frac{\partial E}{\partial V_j} \frac{\partial V_j}{\partial w_{jk}} = \eta \sum_i [D_i - O_i] g'(h_i) W_{ij} g'(h_j) I_k$$

Figure 2-3: Equations for weight updates, per pattern. Note that connections between the input and hidden units are not affected by the errors from the reconstruction layer. Notation adapted from [Hertz *et al.*, 1991].

inputs which are constant during training and testing do not cause problems; they are used by the network as bias inputs.

Both IRRE, and this modified IRRE, are related to auto-encoding networks and principal components analysis (PCA). In auto-encoding networks, the output is trained to reproduce the input layer [Cottrell, 1990]. The hidden layer, which is usually used as a bottleneck, captures important features for reconstructing the input. It has also been shown that under certain conditions, auto-encoding networks will capture the principal components of the inputs [Baldi & Hornik, 1989][Kramer, 1991]. The difference between auto-encoding networks and the networks employed in this study is that the networks used here were not trained to reproduce the input layer accurately; they were trained to perform well on a specific task. The portions of the input which can be reconstructed accurately are the portions of the input which the hidden unit activations have encoded to solve the task. Since only a fraction of the features/inputs may be important for the task, it is difficult to focus attention on task-specific features with auto-encoder networks or PCA. A detailed example of this will be presented in the next chapter.

Although IRRE provides a method to determine which portions of the input the network finds relevant, a notion of time is necessary in order to focus attention in *future* frames. Instead of attempting to reconstruct the current input, the network is trained to predict the *next* input, see Figure 2-4. The prediction is trained in a supervised manner, by using the next set of inputs to appear in the time sequence as the targets. The targets (the next inputs) may contain noise or extraneous features. However, since the hidden units only encode information to solve the task, the network will be unable to construct the noise or extraneous features in its prediction.

2.1 Using the Differences Between Expectation and Realization

To this point, a method to create an expectation of what the next inputs will be has been described. There are two fundamentally different ways in which to interpret the difference between the expected next input and the actual next input. The first interpretation is that the difference between the expected and the actual input is the point of interest

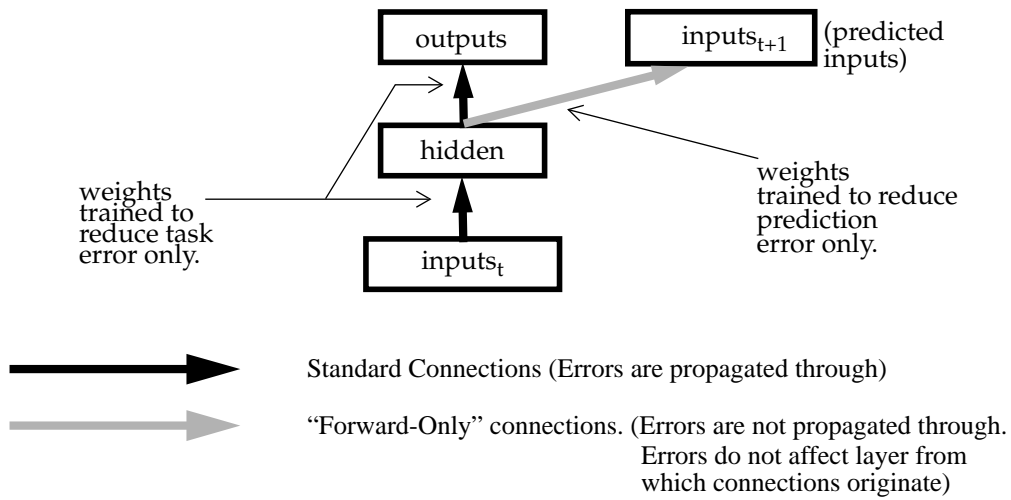


Figure 2-4: Using the activation of the hidden layer to predict the next inputs. The weights between the input and hidden layers are trained to only reduce task error, *not* prediction error. Extra hidden layers for prediction can be used.

because it is a region which was *not* expected. This has applications in anomaly detection, or in the analysis of visual scenes in which the object of interest is moving across a stationary background. This use of expectation will be returned to in Chapter 5, in which detection of anomalies in the plasma-etch step of wafer fabrication is explored.

In the second interpretation, the difference between the expected and actual inputs is considered noise. This interpretation is used throughout the rest of this chapter, and throughout Chapter 3. Processing should be de-emphasized from the regions in which the difference is large. This makes the assumption that there is enough information in the previous inputs to specify what and where the important portions of the next image will be. As will be shown in the tasks described in Section 4 and again in Chapter 3, this method has the ability to remove spurious features and noise.

At this point, we have described a method to compute task-specific expectations. These expectations can be used in a variety of manners. How these expectations are used is dependant upon the task for which they are used.

3. USING EXPECTATION TO REMOVE IRRELEVANT OR DISTRACTING FEATURES

The computed expectation, described above, can be used in a variety of ways. It can be used as extra inputs, or it can be used to filter the current inputs. As mentioned in Section 1, it is necessary to define the role of expectation for the particular application. The application explored in this chapter is lane marking detection for autonomous navigation. For this task, extraneous features should be removed, and the lane marking emphasized. Therefore, expectation will be used to filter out the unexpected features. In visual tasks, such as this one, filtering can be done at a very low level, for example on a pixel-by-pixel basis¹. This type of low level filtering is explained in the remainder of this section. Alternatively, an option which was not explored here but is open for future research, is filtering on an object level. If object detection is incorporated, entire objects can be (de-)emphasized in the input image.

The expectation which was derived in the previous section can be used to filter noise from the current input (see Figure 2-1). For example, in a system which uses images scaled between -1.0 and +1.0 as input, a saliency map is created by using the absolute differences between the expectation of input image_{t+1} (derived from input image_t) and the actual input image_{t+1}. This difference image is scaled to the range of 0.0 to 1.0 with smaller differences closer to 1.0. The result is the saliency map. In order to emphasize these regions for the input, the saliency map is multiplied, pixel by pixel, with input image_{t+1}. The results after multiplication are used as inputs for time t+1. This has the effect of lowering the activation (towards 0.0) of the portions of the input which do not match the expectation. The portions of the input which match the expectation are left unaltered.

The filtering procedure described above works well when the background of the image is a value close to 0.0. As will be shown in the lane-marking detection task in Section 4, this effectively eliminates distractions from the portions of the image in which the road appears, since road pixels often have intensities close to 0.0. The limitations of these

1. This can be considered to be *pre-attentive*, or *early filtering*, as filtering of the image at time (t) is done before the contents of the image are analyzed [Hoffman, 1986]. However, deciding what should be filtered is done after the image (at time t-1) is analyzed.

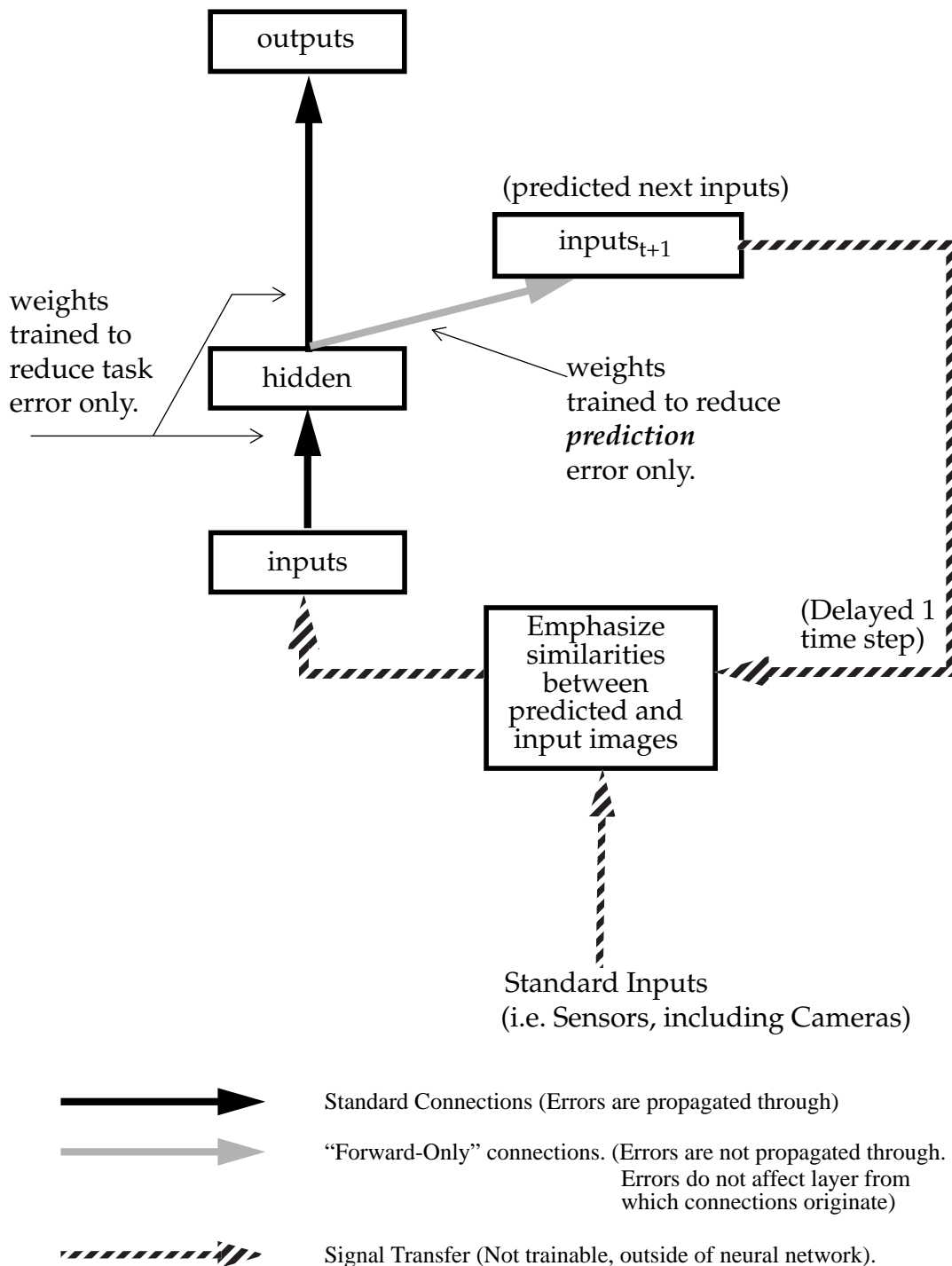


Figure 2-5: Prediction of next inputs, with feedback. The filtering here is done to emphasize similarities. In Chapter 5, in which these methods are used for anomaly detection, the similarities will be de-emphasized, and the differences will be emphasized.

methods will be discussed in Chapter 3. Chapter 4 presents an alternate filtering method which is useful for visual hand-tracking. Chapter 5 discusses using expectations without explicit filtering.

With feedback to the inputs, the system becomes dynamic. As the training for the task improves, the saliency map becomes more refined, and more of the correct information is used for solving the task. Therefore, the images input to the network later in the training process possess different qualities than those input earlier in training. Because the network is trained to reduce the task error, the hidden representation changes to adapt to the new images. This causes changes in the prediction of the next inputs, and the cycle continues. The cycle ends when either the system reaches a stable state or training is stopped. In the experiments attempted in this thesis, the training always converged quickly. The system can be trained by using the standard backpropagation algorithm, with small learning and momentum rates. Training a network with the saliency map does not require more training data than training a network without one. A requirement for using these methods is that the training data must be maintained in temporal sequence, as predictions are made of the next inputs.

As described earlier, one of the problems encountered in focusing attention using this method is the necessity of determining the features which are important for solving the task before the task can be solved. The difficulty is that since the important features are task-specific, and are developed from the network's hidden layer, the task must first be solved. This problem is avoided in the tasks explored here because some of the images used for training may contain little or no noise. Therefore, a small amount of learning is able to proceed without explicit focus of attention. Once a few of the important features are determined, the system can, in many cases, bootstrap itself. Of course, if there is too much noise in the training data, this bootstrapping may not work. An interesting possibility, however, is that information from a user can easily be incorporated to aid in focusing attention. Incorporating *a priori* knowledge is explored in Chapter 4.

It is interesting to note that in this model it is important that the prediction of the future state *not* be too accurate. If the prediction matched the next image exactly, the irrelevant features in the next image would also be reconstructed. Although the network is trained

to predict future inputs with example training images which may contain irrelevant features, the network is *not* able to reproduce the irrelevant features, due to the hidden layer's limited capacity, the task-specific hidden units, and the task-specific nature of training.

3.1 Relations to Other Recurrent Neural Networks

The use of the feedback connections proposed are related to other recurrent neural networks [Jordan, 1989][Stornetta, 1988][Mozer, 1989]. At time t , the Jordan network uses feedback from the output units of time $t-1$, combined with feedback from the context units at time $t-1$, to create new context units. These units are used as additional inputs in the current time-step. The Stornetta architecture uses the context units as a preprocessor of the input units [Hertz *et. al*, 1991]. The context units are arranged with one-to-one connections with the inputs, and have feedback connections from themselves, which carry activation from the previous time step. The Mozer architecture is similar to the Stornetta, except that the connections context and input units are fully connected, and the self connections from the context layer are trainable. See Figure 2-6.

There are several important distinctions between the Jordan, Stornetta and Mozer architectures and the one used in this paper. The first is that the Stornetta and Mozer architec-

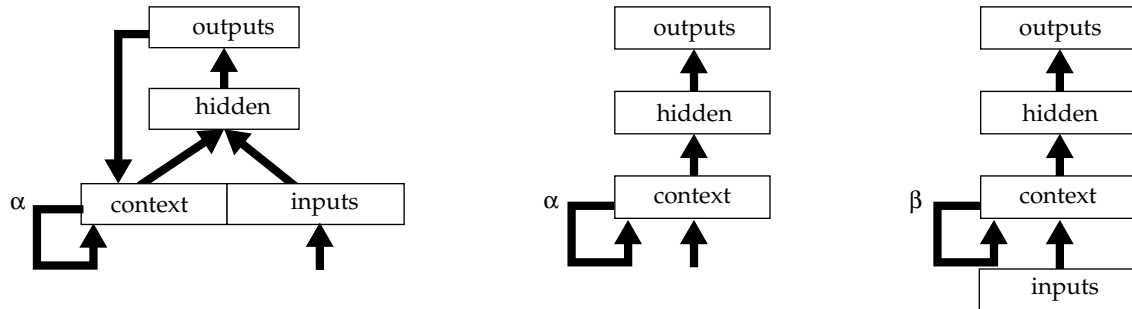


Figure 2-6: (Left) Jordan (Middle) Stornetta (Right) Mozer architectures. Where $\alpha < 1$, β is trained.
From [Hertz, *et. al*, 1991].

tures use context units which are free to form arbitrary representations. In the architecture presented here, the feedback is from units which have a very defined task. The feedback is the prediction of the next inputs. Also, in the implementation described in this chapter, the feedback is multiplicative, and acts like a filter, unlike the Jordan networks.

Another large difference is that in the networks used here, the feedback units are explicitly trained in a supervised manner. In architectures like Stornetta and Mozer, the context units are trained in a manner similar to the training of the hidden units. The representations formed are created to reduce the error in a subsequent output layer.

Finally, the last difference is in the problems which these architectures are designed to address. Most of the recurrent networks which have been explored in the literature have attempted to address the problem of sequence recognition and reproduction. The architectures used in this thesis are not suited to these tasks since the feedback is restricted to be the prediction of the next inputs rather than arbitrary state sequences.

4. EMPIRICAL RESULTS: LANE TRACKING

One of the principle motivations for creating an algorithm which can focus attention is to perform visual processing in cluttered scenes. A real-world application which requires such attention focusing is autonomous road following. In this domain, the goal is to control a robot vehicle by analyzing the image of the road ahead. The direction of travel should be chosen based on the location of important features like lane markings and road edges. This is a difficult task since the scene ahead is often cluttered with extraneous features such as other vehicles, pedestrians, trees, crosswalks, road signs and other objects that can appear on or around a roadway. For the general task of autonomous navigation, these extra features are extremely important. However, for the restricted tasks of staying in the current lane or lane marking detection, many of these features can be distracting. While we have had significant success on the road following task using simple feed-forward neural networks to map images of the road into steering commands [Pomerleau, 1993], these simple methods fail when presented with cluttered environments like those

encountered when driving in heavy traffic, or on city streets.

4.1 The ALVINN Road Following System

ALVINN (Autonomous Land Vehicle in a Neural Network) is an artificial neural network based perception system which learns to control CMU's Navlab vehicles by watching a person drive [Pomerleau, 1993] (see Figure 2-7). ALVINN's architecture consists of a sin-



Figure 2-7: CMU Navlab-5 Vehicle.

gle hidden layer backpropagation network. The input layer of the network is a 30x32 unit two dimensional "retina" which receives input from the vehicle's video camera. The correct steering direction is determined from the activation of 30 output units (see Figure 2-8 (left)). The output units create a gaussian centered around the correct steering direction (see Figure 2-8 (right)). If the Gaussian is centered around unit 1, this indicates the vehicle should make a sharp left, if the center is around unit 30, the vehicle should make a sharp right. To teach the network to steer, ALVINN is shown video images from an onboard camera as a person drives. For each image, it is trained to output the steering direction in which the person is steering.

Recently, there has been an emphasis on using the ALVINN system as a driver warning device. In one of the proposed uses of this system, the model will warn drivers if they begin to drift over lane markings (indicating that they may be entering a lane with on-

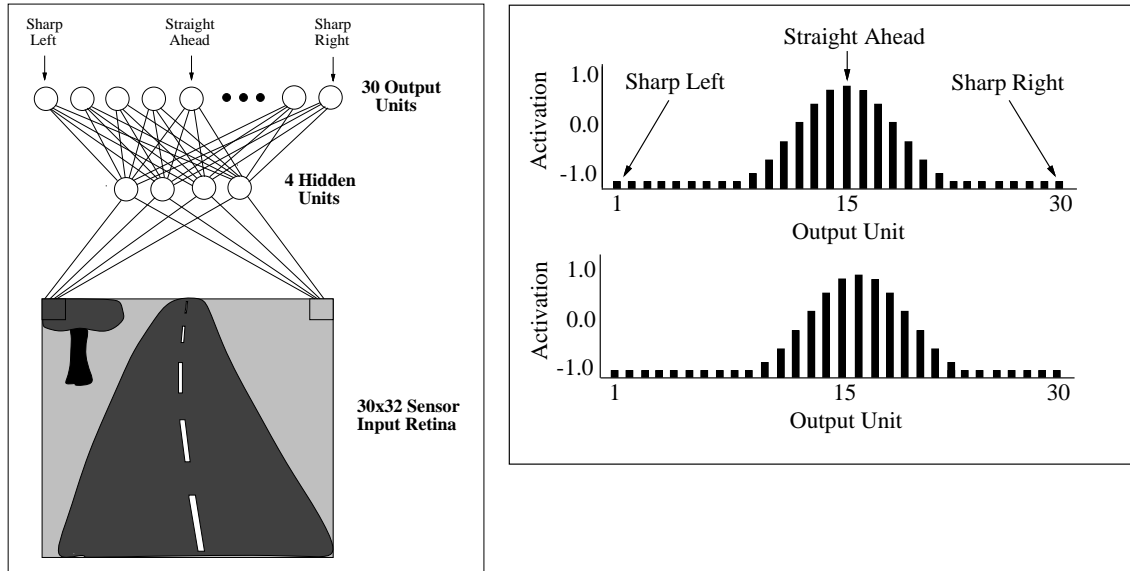


Figure 2-8: (Left) ALVINN architecture, (Right) Gaussian Output Representation for the 30 output units, Top: Straight Ahead, Bottom: Slightly to the right. (from Pomerleau [1993]).

coming traffic, leaving the road, etc.). The system must be robust with respect to other road features, such as road and off-road boundaries, cars, or other lane markings. Experiments for this task are described in the next section.

4.2 Experiment Details

The purpose of using a saliency map within this domain is to eliminate features of the road which the neural network may mistake as lane markings, and therefore output an incorrect steering direction. Training the network to solve the task by focusing on the important regions of the scene is described below.

Approximately 1200 images were gathered from a camera mounted on the left side of the Navlab 5 test vehicle, pointed downwards and slightly ahead of the vehicle. The images were gathered sequentially, at approximately 4-5 images per second. The car was driven through city and residential neighborhoods around Pittsburgh, PA. The images were sub-sampled to a 30x32 pixel representation. From these reduced images, it is possible to steer the vehicle. However, in order to quantify the results with and without the use of the

saliency map, an alternate, closely related task was chosen. In each of these images, the horizontal position of the lane marking in the 20th row of the input image was manually identified. The task is to produce a gaussian of activation in the outputs which is centered around the horizontal position of the lane marking in the 20th row of the image, given the full 30x32 pixel image as input. Sample input images and target outputs are shown in Figure 2-9.

In this task, it is vital to focus on only the relevant portions of the input. If all the input are used, the artificial neural network can become confused by road edges, as shown in Figure 2-9a; by extraneous lane markings, as shown in Figure 2-9b; and reflections on the car itself (since the camera was located on the side), as shown in Figure 2-9d and e.

4.2.1. Training Specifics

In training the network, there are several problems which must be addressed. The first is that there is only a limited amount of training data. As described in [Poggio & Beymer, 1996] and [Girosi, Jones & Poggio, 1995], one of the key problems in many real-world learning-from-example schemes is the training set size. Assuming that the driver has directed the car well, the center line has probably stayed within a small region of the input image. Therefore, the network has not been trained to recognize lane markings out-

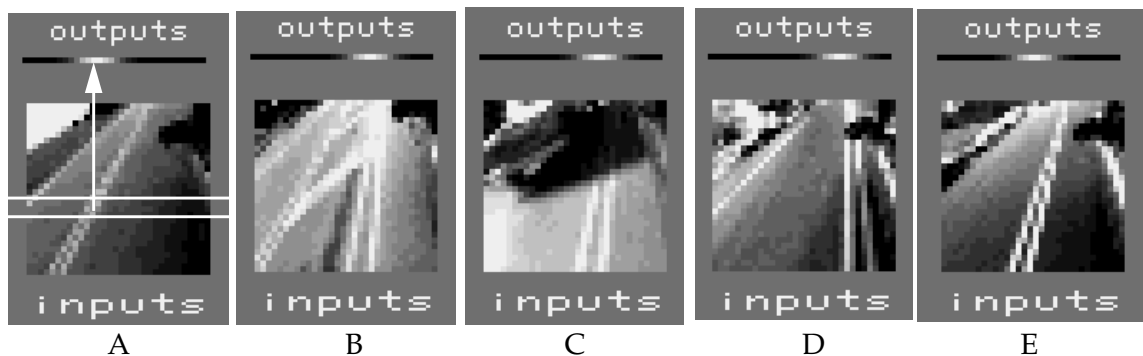


Figure 2-9: Five sample input images and target outputs. Image A shows the region from which the lane marking was manually selected. In image B, there is an extra lane marking. In image C, the lane marking is not completely visible. In images A and D, road edges appear similar to lane markings. In D and E, reflections from the car and road edge may be confused for lane markings.

side the middle regions of the image. The second problem is that because the prediction task attempts to forecast future inputs, it is important to prevent the network from memorizing the image transitions in the training set. For example, had the driver chosen a slightly different action, the location of the important features in the next set of inputs may have been different.

In order to alleviate the problems with the training set, the training set was augmented as follows. In training, extra input images were created by translating the original images to the left or right by up to 5 columns; the unknown regions in each row were filled in with the last known pixel value of each row. The output was also translated either to the left or right by the same amount as the image. This translation yields usable images because the camera is pointed downwards. This is a standard method of generating training data for situations which the driver may not have encountered, such as cases in which the lane marking is at the left edge of the input. If the camera had been pointed more ahead of the vehicle, a more sophisticated image transformation would have been required to maintain the correct perspective, as was used in [Pomerleau, 1991].

The sequential nature of focusing attention dictated that these images could not simply be added to the training set. For example, an image at time t , which is translated 3 columns to the left, should not be followed by an image at time $t+1$ which is translated 3 columns to the right. If it were, the important features would jump a large distance, and this is unlikely to happen in practice. To avoid this problem, the expanded training set is used in the following manner: the image at time step $t+1$ is chosen at a random translation which differs by, at most, ± 1 columns from the translation of the image at time step t . This ensures that there are no artificially generated large jumps of the important features between consecutive time steps. As the network is trained through many passes through the training set, images are seen with different translations.

The training data used in these experiments were collected from a single driver. Since this system is to be used in a wide variety of situations, the system must be made robust. For example, in many images, driving straight or steering slightly to the left or to the right may be an acceptable maneuver. To ensure that the network learns these possible transitions, for each input image, many potential “next input” images are created. The network

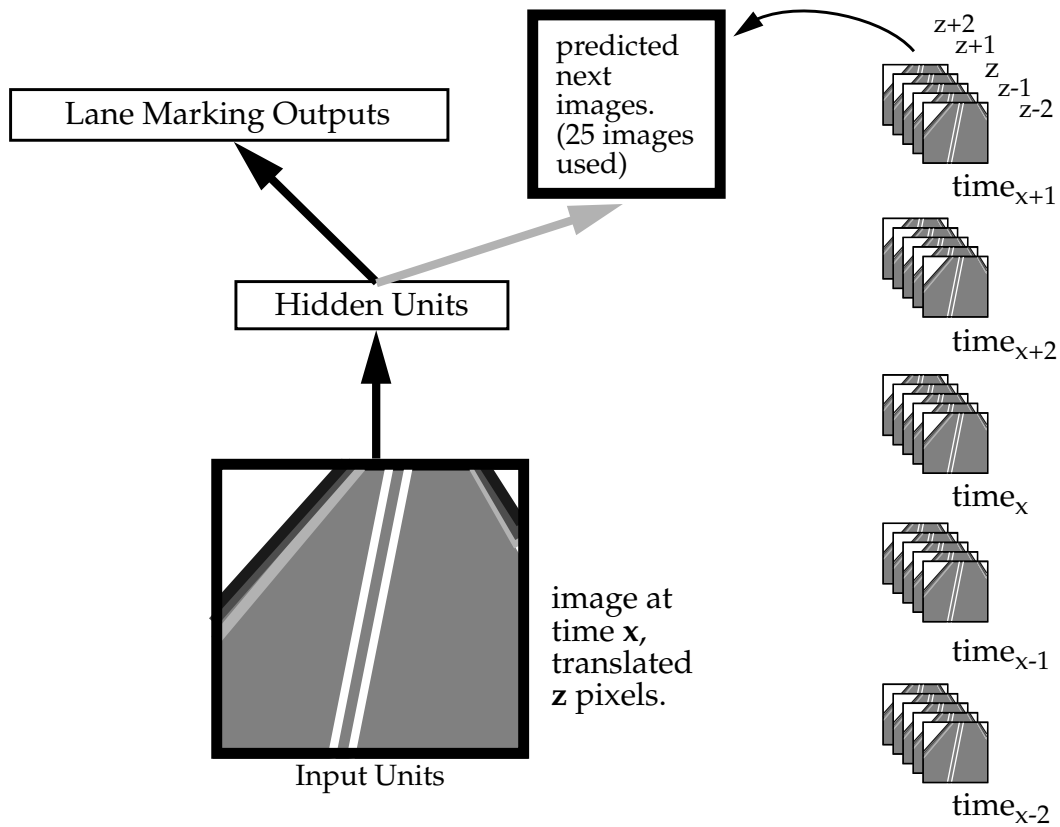


Figure 2-10: Training the network to account for multiple responses. Images for prediction are created by translating to the left and right ± 2 columns of the input image's translation. Images from current, future, and previous time steps are also used.

is trained to predict all of these synthesized images. These images are created by translating images which are temporally close to the current image, to the left and right of the current offset in the current image (in a manner similar to the one described in the previous paragraphs). The errors between the predicted next state and all of these images are used for training the network, since any of the images are reasonable expectations for the next input (see Figure 2-10). In the experiment presented here, 25 “next input” images (5 time steps \times 5 translations per time step) were paired with each input image. One interesting aspect of the training data in this domain is that images from recent *previous* time-steps can be used for this procedure, since many of the features remain consistent for short periods of time. Previous images are unusable for training next state predictions in many domains, as will be shown in Chapter 3.

The first 800 images were used for training. The last 400 images were used for testing. Both training and testing images were used with translations. Overtraining was not encountered, possibly because of the use of random translations through training which kept the training set changing.

4.2.2. Results

After training in the manner described above, the results of this experiment were very promising. The lane tracker was able to remove various types of distracting noise from the images. Figure 2-11 shows 4 image triplets. The left image of each triplet displays the original, unfiltered, input image_t. The middle image is the predicted image_t, made by the network with input of image_{t-1}. The right image is an image which has been filtered, pixel by pixel, by the absolute difference between the original image_t and predicted images_t (it has been filtered by the saliency map). The larger the difference between actual and predicted image, the more the pixel's activation is reduced. In triplet A, the edge of the road is bright enough to cause distractions. In B, the two lane markings may confuse the lane tracker, and cause it oscillate between the lane markings. In C, a passing car's reflectance may be confusing. In each of these three triplets, the confusing attributes are removed from the image (shown in the right image of each triplet). Triplet D shows the limitations of the saliency map; although it is able to eliminate the majority of the distracting lane markings, not all of them can be removed. Nonetheless, the most distracting lane marking was removed, and the ANN with the saliency map was not confused by those which remained.

The expectations which are displayed in column 2 of Figure 2-11 show that the expectation of where to find the lane marking and the road edges is not precisely defined. This is due to the training method, which attempts to account for the many possible transitions from one time step to the next. This is vital to ensure that the important features do not fall outside of the expected area. If the important features fall outside the area in which they are expected, they may be de-emphasized in processing.

Quantitatively, the performance of the lane-tracker with the saliency map, measured by the difference between the estimated and actual position of the lane marking, revealed an

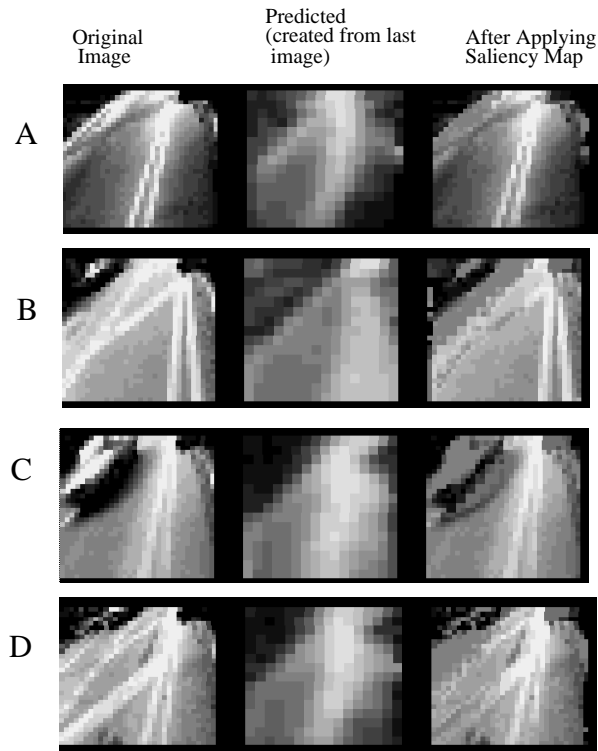


Figure 2-11: **Left:** original input image_t. **Middle:** predicted image_t, made by the network with input of image_{t-1}. **Right:** right image is the image which is filtered, pixel by pixel, by the difference between the original image_t and predicted images_t. The larger the difference between actual and predicted image, the more the pixel's activation is reduced to 0.0 (a value of intermediate gray). The right image is used as the input to the neural network. The images are scaled between value of -1 and +1. Note that these images are shown without translation.

average improvement of 20% over the lane-tracker without the saliency map. This improvement was revealed over multiple runs, with random initial weights in the ANN and different random translations chosen for the images during training. The improvement was not greater because many of the images in the test set do not contain noise. In the images without noise, a standard ANN can be used to accurately estimate the lane marking position. Nonetheless, in order to maintain a user's trust in a driver warning system, it is crucial that false alarms are minimized. Further, since the system is designed

to take control of the vehicle in hazardous situations, under no circumstances may the system be distracted by spurious lane markings or similar appearing features.

In terms of filtering, there are important differences between this application and many other studies conducted in non-visual domains. First, the inputs in this domain are highly redundant. Therefore, the filtering does not have to be perfect. If important pixels are accidentally removed, there may be many other pixels which contain enough information to recover. Second, it is important to eliminate as many of the irrelevant pixels as possible, since they may contain information which directly conflicts with the correct information. Third, this task emphasizes the importance of dynamically re-evaluating the relevance of each of the inputs. As the images change, the relevant pixels change.

One might be tempted to think that since the actual next image is filtered based upon the differences between it and the expected image for that time-step, that it should be possible to just use the expected image as input, and ignore the actual image (provided that the predictions are accurate). However, this is not the case. As can be seen in Figure 2-11, the expected images predict *a region* in which the lane-markings will appear. The actual lane-markings, which are located within this region, give the basis for the prediction of the next lane-markings. This new information is incorporated to make accurate future predictions. If only the expected image had been used, the expected image would continually get more diffuse, until all information is lost. This is somewhat analogous to Kalman Filters, in which each measurement is used to revise the prediction of future measurements; the longer the predictions are continued with no readings, the larger the error in prediction will be. Additionally, the actual input image's activations are reduced to the *background intensity*, which may not match the activations in the predicted image.

It should be noted that selective attention can also be used when the regions of interest are spatially discontinuous over time. There is no prerequisite for smooth transitions of focusing regions, as is commonly accepted for biological systems [Umla, 1988]. Examples of this are explored in depth in Chapter 3.

5. OTHER NETWORK ARCHITECTURES FOR FOCUSING ATTENTION

There are many alternatives to the implementation of the selective attention mechanisms described in this chapter. Three interesting options which were not implemented in these experiments are described below. The first two correspond closely to the decisions made for the selective attention mechanisms described here. The third is an option for future work which can be used with and without the selective attention mechanisms.

In the implementation described in this chapter, the prediction was based on a linear combination of the activation of the hidden units. This is not a necessity for the mechanisms developed in this thesis. If more than a linear transform of the hidden units is necessary, a hidden layer dedicated to the prediction task can be added (see Figure 2-12(bottom)). Also, if multiple hidden layers are necessary to solve the task, the prediction layer can be connected to all of the hidden layers (see Figure 2-12(top)). The crucial detail is to ensure that the errors which are propagated back from the prediction do not effect the representations which are developed in the hidden layers for solving the task.

A decision which could have been made differently for this task is when to start the feedback from the prediction layers. During the initial phases of training, when the prediction is little more than random noise, filtering based on prediction can severely degrade input images. Performance improves once the network can make at least crude predictions of the next state. Initially, predictions are made through the use of the bias unit to predict the “average” next state. Eventually, once task-specific features are developed in the hidden units, more accurate predictions are made. One way to both reduce the training times and to potentially improve performance is to begin the feedback after some initial learning of the prediction has progressed. This will avoid the early stages of training when the inputs are filtered by essentially random feedback.

Finally, the last option may be useful both with and without the selective attention mechanisms. One method to enhance the amount of information that is preserved from one input presentation to the next, is to not only feedback the prediction of the next state (as is currently done), but also to feedback the outputs and/or the hidden layers of the net-

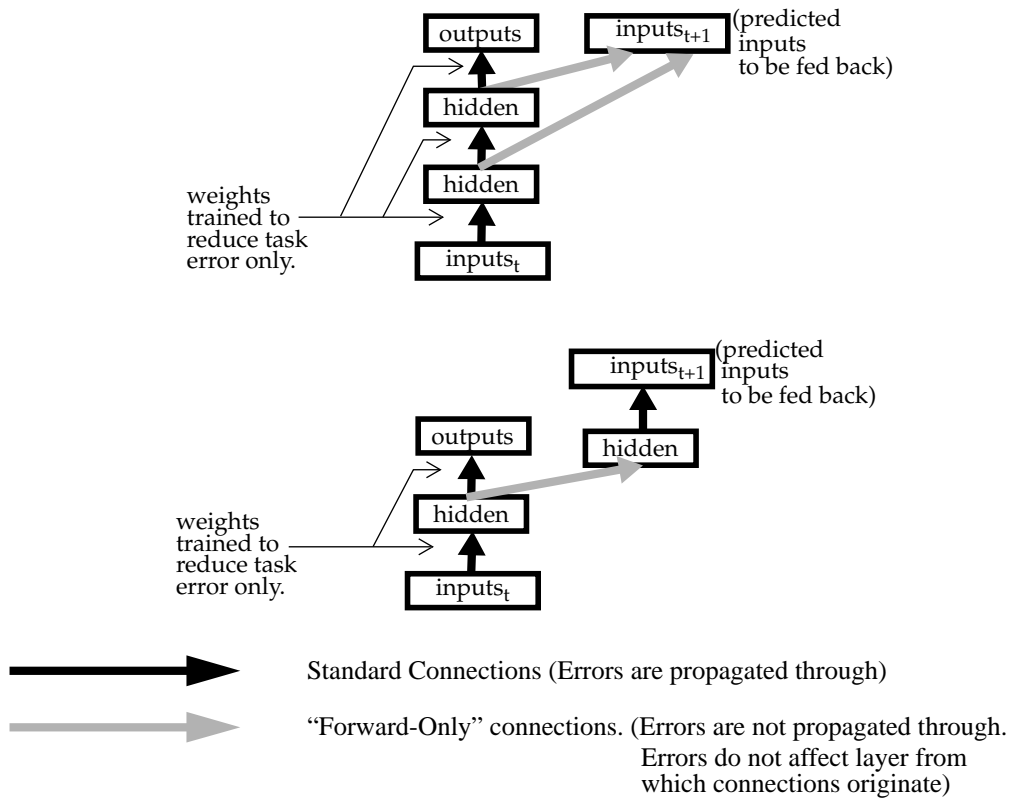


Figure 2-12: Top: Multiple hidden layers for solving the main task – all are connected to the prediction outputs. Bottom: A hidden layer is used for solving the prediction task. Because of the “forward-only” connections in each of the networks, the errors from the prediction task do not affect the development of the hidden layers used for the main task.

work from the previous time step. This is similar to the Jordan network [Jordan, 1989]. Although this information cannot be directly used for focusing attention in the same way as the saliency map, it may be useful in determining the correct steering direction when there is little information in the current input image. It will allow the network to remember its previous action.

6. SUMMARY

In this chapter, a method for focusing attention on the portions of the input which are important for completing a particular task was presented. The artificial neural network

used in this study was able to avoid distractions by focusing attention on only selected portions of a scene. The resulting network can be trained by using the standard back-propagation learning algorithm. Extensions to the presented architecture can use methods of encoding task context (such as the Jordan, Stornetta and Mozer architectures) in addition to the attention mechanisms.

Unlike many of the previous studies which use neural networks to predict future states, this work has presented an algorithm which relies on the *limited* accuracy of the neural network's future state prediction. The premise of this algorithm is that future event prediction cannot be perfect. In particular, the network cannot accurately predict the future states when the future states contain noise or spurious features. By analyzing the difference between what is expected in the next time step and what is actually present in the next time step, it is possible to estimate which of the features in the input retina are noise, and which are important to completing the particular task. Based upon the features that were found, new, revised, expectations can be formed for subsequent time-steps. This update of predictions is related to Kalman filters, in which the previous estimates of future images are constantly revised as more information is given to the system. The relation will be reviewed in detail in Chapter 7. In the techniques presented here, the relevant attributes (those which are encoded to solve the task), the transition models of the relevant attributes, and the noise models, are automatically determined.

This chapter has demonstrated the applicability of the presented approach on a real-world application: lane-marker tracking for autonomous road following. The resulting system is robust to many forms of distracting features in the inputs. The algorithm is able to avoid being misdirected by extra lane markings, passing cars, and other potentially confusing features by selectively emphasizing the portions of the input which match closely to the prediction of relevant inputs.

CHAPTER 3

UNDERLYING ASSUMPTIONS FOR TASK-SPECIFIC EXPECTATION-BASED SELECTIVE ATTENTION

In this chapter, some of the underlying assumptions of the prediction based selective attention methods are examined. Because the focus of attention methods explored in this thesis rely heavily on the ability to predict the next input scene, this chapter explores the requirements for making accurate predictions. In particular, this chapter shows, through a series of carefully constructed synthetic examples, the type of information which must be present in a neural network's hidden layer for the prediction mechanisms to perform correctly.

1. OVERVIEW

Chapter 2 gave a method to create task-specific predictions of the next time step's inputs and a filtering procedure to use these predictions. The filtering procedure scales the next time-step's inputs towards the background color in proportion to how much they differ from the predicted inputs. In this chapter, we use these same prediction and filtering procedures in a variety of synthetic tasks. These tasks were created to illustrate the type of information which must be present in a neural network's hidden layer for accurate predictions. In later chapters, we examine alternate filtering procedures.

The basis for the experiments in this chapter is the detection of a cross in an image. In this task, a cross appears in one of 16 positions in the input image, and the desired output is the position of the cross. In each image, the cross appears in different positions. The task is complicated by the appearance of distracting crosses in the input. Four sets of experiments are explored in this chapter; each is designed to elucidate a different aspect of expectation-based selective attention:

1. *Discontinuous Cross Detection*: In many traditional vision-based tracking systems, it is assumed that the salient regions are spatially continuous. First, this section shows that spatial continuity is not required for these methods. Second, the methods described in the previous chapter de-emphasized features which were not predictable; this section shows that even distractions which are predictable can be removed, as long as they are not related to the task.
2. *What is encoded in the hidden layer? State, Not Position*: This section shows that a network does not necessarily encode the position of the cross in its hidden layer, but rather an abstract transformation of the input which can be converted into position. In the general case, this is important when different inputs (representing different states of the underlying process) correspond to the same output. If only information regarding the output was encoded in the hidden layer, these different states could not be differentiated by analyzing the hidden layer.
3. *What information is necessary to make accurate predictions? Combined Recognition and Detection experiments*: To explicitly control the type of information encoded in the hidden layer, a prediction task which is based on two features in the input was devised. To see the effects of the contents of the hidden layer on prediction ability, training runs with supervised signals for each of the features, individually and together, were conducted.

4. *Why not use Principal Components Analysis (PCA)?* Autoencoding networks, or PCA, are often used for anomaly detection tasks. However, these techniques are not task specific. Both of these techniques encode all portions of the input without regard to whether the encoded information is necessary for anomaly detection. Therefore, they are not efficient when the output of the network is determined by a small percentage of the total available inputs.

This chapter contains the most numerous and detailed experiments in this thesis. Although a thorough understanding of all the experiments in this chapter is not necessary to understand subsequent chapters, the lessons learned from the experiments are important for understanding the task-specific expectation-based selective attention methods introduced in Chapter 2. The main lessons are summarized below:

1. Unlike in many vision-based tracking systems, salient regions do not have to be spatially continuous through time for this system to work. See Section 2.
2. Even if there are predictable distractions, because of the task-specific training, these distractions can be de-emphasized. See Section 2.2.
3. If too small a hidden layer is used, too much information may be lost due to compression. A small hidden layer is particularly a problem when different inputs (representing different states of the underlying process) correspond to the same output. See Section 3.
4. In some cases it will only be possible to limit the search for an object to a set of locations rather than to a specific location. However, this set can still be used to focus attention. See Section 4.
5. Non-standard network architectures, such as those which use retinal connections from the input to hidden layer, can effectively maintain spatial information about the inputs. See Section 4.
6. Large hidden layers can encode irrelevant information about the inputs. Weight decay and early stopping limit this. See Section 5.
7. Because PCA is not task-specific, it may be inefficient for detecting anomalies in tasks that have only a few relevant inputs and many irrelevant ones. See Section 5.

1.1 The Experiments

For the experiments presented in this chapter, the saliency map is constructed in the same manner as described in the lane-marking task in Chapter 2. Predictions of the next inputs are made based on the network's hidden layer. In the raw inputs, the crosses appear with an activation of +1.0. The background activation is set at -1.0. Note that complete knowledge of the background activation will usually not be possible in real tasks. For example, in the lane-marking experiments in the previous chapter, the road appeared with an activation close to 0.0. However, the rest of the image had a different average activation. To examine the effects of this type of filtering when the background is not known, experiments were done by scaling the next time-step's inputs *towards 0.0* in proportion to how much they differ from the predicted inputs. This has two effects: (1) when markers are to be de-emphasized, they will only fade, rather than disappear from the input. (2) When incorrect predictions are made, they will appear in the input image as ghost features, as discussed in Section 4.4. However, as long as correct predictions are made, this will not hurt performance. The filtered inputs are used as inputs to the network. As in the lane-tracking experiments, the main task outputs and the prediction outputs are trained simultaneously.

2. DISCONTINUOUS CROSS DETECTION

The cross detection problem contains many of the difficulties of the road following task described in the previous chapter, in terms of changing relevance of the inputs and focusing attention. However, it is a simpler domain, and lends itself to easier interpretation and more controlled analysis. This problem also serves to demonstrate the saliency map's ability to work in domains in which the relevant inputs are spatially discontinuous over consecutive frames. This type of model is useful when considering tasks in which the presence of one feature triggers an expectation of another feature, in the next time step, in a different location of the input.

In this task, a cross (“+”) of size 5x5 appears in a 20x20 grid. There are 16 positions in which the cross can appear (see Figure 3-1). The cross moves according to the transition rules shown in Figure 3-1b. The object of this problem is to reproduce the cross in a

smaller 10x10 grid, with the edges of the cross extended to the edges of the grid, as shown in Figure 3-1a. The task is complicated by the presence of randomly appearing noise. The noise is in the form of another cross which looks the same as the cross of interest, but its movement does not follow the transition rules. In this task, the network must be able to distinguish between the real cross and distractor crosses. This is only possible with knowledge of the previous image(s). The prediction of future images is used to discover the transition rules and to focus on the next location in which the cross will appear. If the correct location is not focused upon, the contents of other locations can confuse the neural network, thereby degrading performance [Baluja & Pomerleau, 1994].

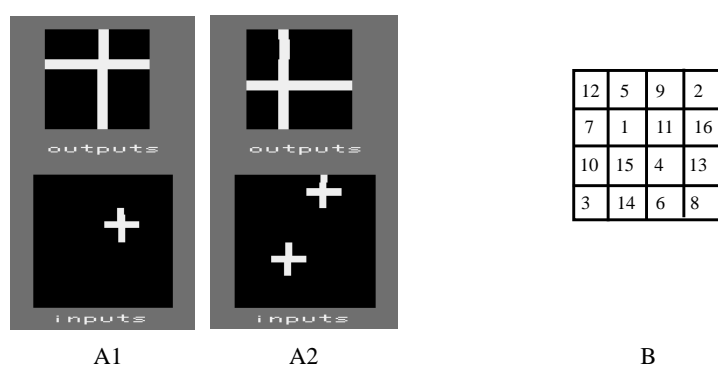


Figure 3-1: (A1) Task with no distractions, (A2) with a distractions. (B) Transition rules.

2.1 Experiment 1: Random Noise

For these experiments, a fully connected network with a single hidden layer with 12 hidden units was used. The network training was completed in exactly the same manner as described before in the lane-marker tracking task of Chapter 2 (without the measures taken for augmenting the training set). Several training variants were tested: with and without a saliency map, and with and without noise in the training set. The network with the saliency map must learn the transition matrix given in Figure 3-1b to effectively focus attention. In this task, either 1 noise or 2 noise markers are present. '1 noise' means that there is a 50% chance of a noise element occurring in the image, '2 noise' means that there is a 50% chance of another noise element occurring, independently of the appearance of the first noise element. The positions of the noise elements are determined randomly.

When neither the training nor test sets contain noise, the saliency map will not provide any benefit. This is expected as no focus of attention is needed to solve the task. When noise is introduced into either training or testing, the saliency map provides an effective means to focus attention (see Table I). The network without the saliency map, trained *with* noise, and tested *without* noise, cannot perform well. *This indicates that the presence of noise in the training set prevents accurate learning.* As expected, when the testing set becomes harder, by introducing noise, the performance further degrades.

Table I: Output Error: Summed error of all outputs, averaged over 10,000 examples.

Training Set	Testing Set			
	0 Noise		1 Noise	
	Error	Percent Improvement with Saliency Map	Error	Percent Improvement with Saliency Map
0 Noise (No Saliency)	0.7	0%	13.4	30%
0 Noise (Saliency)	0.7		9.4	
1 Noise (No Saliency)	3.2	84%	15.0	93%
1 Noise (Saliency)	0.5		1.0	

When the training or testing sets contain noise, the network with a saliency map works better than the network trained without a saliency map. Further, the presence of noise in the training set does not hurt the performance of the network with a saliency map.

It is interesting to note that even when the networks are trained without noise, using a saliency map improves performance when testing with noise. As expected, this performance improvement is not as great as when the networks were also trained with noise. For example, a network trained with only one marker ever appearing in the inputs will not necessarily be able to handle inputs which contain two markers, even if one of the two markers is de-emphasized, since the overall activation of the inputs and the discrimination thresholds will not be set correctly. The same is true when the number of noise markers increases beyond the training set. The results in Table II show the summed error over all of the outputs when 2 noise elements were used for testing. The worse performances of networks trained both with and without a saliency map is expected since the

testing set is drawn from a different distribution than the training set.

Table II: Performance Degradation with Extra Noise. Summed error of all outputs, averaged over 10,000 examples.

Training Set	Testing Set	
	2 Noise	
	Error	Percent Improvement with Saliency Map
0 Noise (Saliency)	21.7	18%
0 Noise (No Saliency)	17.9	
1 Noise (Saliency)	22.8	59%
1 Noise (No Saliency)	9.4	

In Figure 3-2, a segment of a typical test run is shown. In the figure, the inputs, the predicted and target outputs, and the predicted and target next inputs are shown. The target prediction is just a reduced version of the unfiltered next input image. The input size is 20x20, the outputs are 10x10, and the saliency map is 10x10. The saliency map is scaled to 20x20 when it is applied to the next inputs. Note that in the inputs to the network, one cross appears much brighter than the other; this is the effect of the saliency map – it suppresses the distracting cross.

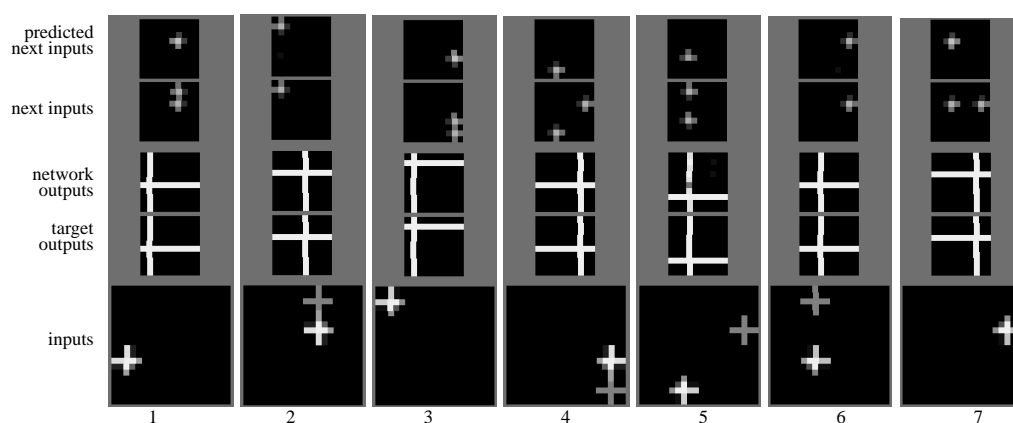


Figure 3-2: A typical sequence of inputs and outputs. Note that when two crosses are in the inputs, one is brighter than the other. The “noise” cross is de-emphasized.

In contrast to many standard vision-based tracking schemes, this task showed that spatial continuity through time is *not* required for the transitions to be learned. A more complex task, which also has this property, is one in which the noise marker follows a fixed transition matrix which is different from the transition matrix followed by the real cross. In this revised task, the noise is “temporally structured,” which makes it more difficult for the prediction-based techniques presented in this thesis, since predictions can be made for the noise cross also. Nonetheless, assuming that the two markers are randomly restarted in different locations in the image during training, this approach can be used, without any modifications, to learn the transition rules of the real cross [Baluja & Pomerleau, 1995]. This task is described below.

2.2 Cross Detection with Temporally Coherent Noise

The cross detection task described in the previous section does not require a neural network. Rather, accurate predictions of the next state can be made by counting the number of occurrences of a cross in each position of the next inputs, for each position of the cross in the current inputs. By examining many samples and recording the corresponding next state, the transition rules can be determined. This can be done because the noise markers appear randomly, while the real marker appears only in one location. Of the 16 next location counters, the correct one will be the maximum.

In the previous task, the distracting crosses appeared in random locations. In this task, the distracting cross appears in a pre-defined order, in the same manner as the real marker. The sequence of moves of the noise and real markers are different. The transition matrices are shown in Figure 3-3.

Real Marker:

12	5	9	2
7	1	11	16
10	15	4	13
3	14	6	8

Distraction Marker:

7	3	1	4
16	10	14	12
8	5	2	9
15	11	13	6

Figure 3-3: Temporally coherent noise. Transition matrices for real and distractor markers.

In the current task, for training, both markers are each restarted in different randomly chosen positions in their sequence every 50 pattern presentations. Note that the counting technique described above will still work. Nonetheless, for a neural network, this task is more difficult since for each location there will be two strong competing alternatives for the prediction of the next position. In the first pattern presentation of every restart, there is no noise marker. This initializes the network with the correct marker. If two markers were present in the first image, it would be impossible to determine which marker is the correct one to focus upon. The experiments in this section were done with varying the percentages of pattern presentations which contained noise markers in both the training and testing phases. See Table III.

Table III: Experiments with Temporally-Structured Noise, average error on 1000 examples, summed over all outputs.

Training Procedure		Performance				
Saliency Map	% of Training Examples Containing Noise	% of Testing Examples Containing Noise				
		0%	25%	50%	75%	100%
No	25%	1.8	8.4	14.2	20.1	26.6
Yes		0.3	0.3	0.3	0.3	0.3
No	50%	1.9	8.3	13.8	20.0	25.7
Yes		0.3	0.3	0.3	0.3	0.3
No	75%	2.0	8.3	13.8	19.9	25.6
Yes		0.4	0.4	0.4	0.4	0.6
No	100%	5.5	11.2	16.0	21.5	26.7
Yes		0.5	0.5	0.5	0.7	1.0

The motivation for having the noise marker follow an underlying transition matrix, instead of simple randomly occurring noise, is that it more closely approximates real-world tasks in which distractions may be coherent, structured features or objects which persist through multiple time steps. Examples of this include multiple voices or conversations in the context of speech recognition, or multiple lane markers in the context of lane tracking (as described in chapter 2).

3. ENCODING STATE – NOT POSITION

In the experiments described above, each input position corresponds to a unique output state (one-to-one). However, many tasks may not have this property. Other tasks may have the property that multiple input sets correspond to the same task output (many-to-one). For these cases, it is vital to understand what needs to be encoded in the hidden layer to make accurate predictions of the next inputs. In particular, if the same information is encoded in the hidden layer for different input states, will the prediction task, which is based solely upon the activations of the hidden layer, be able to disambiguate different input states which have the same representation in the hidden layer? When a network is trained to solve the cross detection task, what is encoded in the hidden layer? This section will show the distinction between encoding state and encoding marker position.

For these experiments, a modification of the marker location task, described in Section 2.1, is used. Rather than using the output of the location, or outputs which encode location directly, a target output of N bits is randomly associated with each input pattern (see Figure 3-4). N will be varied through this section to examine its effects of the representations developed in the hidden units. An important feature to keep in mind throughout this section is whether the assignment of bits to each of the sixteen inputs states is unique.

There are 16 unique positions of the real cross in the inputs. If unique output assignments are not used, for example if N is less than 4 ($= \log_2(16)$), then there is no guarantee that the hidden layers will encode unique activations for each of the input patterns. Assuming that we are using a single hidden layer feed-forward neural network, with no skip connections, we know that if the outputs for two patterns are *different*, then the hidden representation for the two patterns must be different. However, if the target outputs are the *same* for different input patterns, and the network classifies them correctly, then it is *not* the case that we can assume the hidden representations are the same for any of the inputs. In fact, the methods employed in this thesis use the fact that the representations will most likely not be exactly the same.

The first experiment that we look at uses $N=5$. For the case of using unique outputs vec-

tors for each of the inputs, it is clear that we can solve this problem, since we were able to solve the original position problem, which only encodes 4 unique bits (16 positions). It is more interesting to see what happens when we try this problem with output vectors which are not unique. We replicate the latter case with $N=3$ and $N=1$. Results for all of these experiments are shown in Table IV, Table V, and Table VI, respectively. The results are the average of at least 9 runs, with random initial weights and randomly assigned outputs of 5, 3, and 1 bits respectively. Randomization of target outputs is done individually at the beginning of each of the 9 runs. Performance is measured in terms of









Marker Position	N (bits)			
	5 Unique	5 Non-Unique	3	1
	10011	11111	001	1
	00001	00001	010	1
	00010	00001	011	0
	10001	10001	100	1
	01110	01110	101	0
	10101	10101	110	1
	01010	01010	110	0
	11111	11111	110	0

Figure 3-4: Sample targets assignments for 8 of the 16 board positions.

Table IV: N=5, 12 hidden units, average SSE on 1000 examples for each output

Training Set	Testing Set					
	0 Noise		1 Noise		2 Noise	
	Error	Improvement w/ Saliency	Error	Improvement w/ Saliency	Error	Improvement w/ Saliency
0 Noise (No Saliency)	13	15% (not significant)	226	35%	353	25%
0 Noise (Saliency)	11		148		263	
1 Noise (No Saliency)	56	98%	252	86%	366	59%
1 Noise (Saliency)	1		34		151	

Table V: N=3, 12 hidden units, average SSE on 1000 examples for each output

Training Set	Testing Set					
	0 Noise		1 Noise		2 Noise	
	Error	Improvement w/ Saliency	Error	Improvement w/ Saliency	Error	Improvement w/ Saliency
0 Noise (No Saliency)	13	23%(not significant)	229	31%	346	23%
0 Noise (Saliency)	10		159		268	
1 Noise (No Saliency)	53	98%	250	80%	355	50%
1 Noise (Saliency)	1		50		176	

Table VI: N=1, 12 hidden units, average SSE on 1000 examples for each output

Training Set	Testing Set					
	0 Noise		1 Noise		2 Noise	
	Error	Improvement w/ Saliency	Error	Improvement w/ Saliency	Error	Improvement w/ Saliency
0 Noise (No Saliency)	11	36% (not significant)	224	44%	352	30%
0 Noise (Saliency)	7		125		245	
1 Noise (No Saliency)	50	98%	249	85%	365	56%
1 Noise (Saliency)	1		38		161	

sum of squares error (SSE).

The most interesting result is found in the case of using $N=1$. Here the target outputs cannot be unique for all 16 patterns. Therefore, it is not guaranteed that the hidden layer will form unique representations for each of the output states. Before the results are examined in detail, it is important to understand whether we should expect any improvement in performance using the saliency map if the representations which are formed in the hidden layer are not unique. Lets examine the worst case, in which the representations formed for all identical outputs are exactly the same in the hidden layer. Assuming a uniform distribution of 0 and 1 outputs, the activation in the hidden layer will lead to predictions of seeing a marker in 8 positions in the next inputs predictions, since 8 of the input states will cause the same representation in the hidden layer. This is in contrast to having a unique representation in the hidden layer, which would lead to a prediction of seeing a marker in only one position in the next inputs. Therefore, in the worst case, we will be able to remove noise markers only 50% of the time – when they appear in one of the 8 locations that were not predicted. Unfortunately, the prediction strength is proportional to the number of times the prediction is made correctly (remember, the prediction is trained by standard neural network training methods) – and it will be incorrect in 7 out of the 8 locations which it predicts. The network, in fact, averages the predictions for each state with the same hidden unit representation. Thus, the predicted image will be dim, providing only weak support for seeing a marker in multiple positions.

Surprisingly, the improvement in using a saliency map for $N=1$ outputs is almost the same improvement that we see in the cases in which $N=5$ and $N=3$. Therefore, we expect that the activations in the hidden units must be unique. If we examine the hidden activations for each of the inputs, we see that they are unique. See Figure 3-5.

It should be noted that there is no guarantee that retraining the network with different initial weights will again encode all of this information (although it did for each of the 9 runs attempted). In fact, it is possible to construct a situation in which the network will not encode this information. In the experiments presented to this point, the network was able to encode all of the information because of the large hidden layer (12 units) relative to the task. There was no reason to compress the information from the input layer since

















Marker Position	Desired Output	Hidden Unit Activations (12 hidden units)	Difference from Average Hidden Unit activation of each class. Average RMS difference for each output.	
			avg '0' Class	avg '1' Class
	1		0.49	0.11
	1		0.47	0.11
	0		0.09	0.48
	0		0.09	0.48
	1		0.48	0.11
	0		0.08	0.47
	1		0.49	0.11
	1		0.48	0.11

Figure 3-5: Hidden Unit Activations are shown for 8 of the 16 states. Note that they are different, although many of the desired outputs are the same. 12 hidden units are used. The variability within classes is higher than when only 2 hidden units are used, see Figure 3-6. Intra-class variation is highlighted.

the hidden layer was not a severe bottleneck. However, if the hidden layer is made a bottleneck, the network will be forced to compress the information. To see this effect, experiments with a smaller hidden layer (only 2 units) were performed. The results are shown in Table VII.

As can be seen, when tested and trained without noise, the network does about the same

Table VII: N=1, 2 hidden units, summed error on 1000 examples

Training Set	Testing Set					
	0 Noise		1 Noise		2 Noise	
	Error	Improvement w/ Saliency	Error	Improvement w/ Saliency	Error	Improvement w/ Saliency
0 Noise (No Saliency)	9	0%	221	11%	352	9%
0 Noise (Saliency)	9		197		320	
1 Noise (No Saliency)	57	38%	250	11%	367	8%
1 Noise (Saliency)	35		222		338	

as a network with a larger hidden layer, indicating that the main task can be solved reliably. However, the prediction task is unable to work as well, suggesting that the similar output states have similar hidden representations. If we look at the activations (Figure 3-6), we see that the patterns which have the same outputs have nearly identical hidden unit representations. Theoretically, since the hidden activations are not *exactly* the same, a network should be able to discriminate between them (possibly given more units/layers).

These experiments reveal a general principle for using the selective attention methods. Tight compression of the information from the input layer into the hidden layer is undesirable in tasks in which many different inputs should be classified to the same output. The tight compression forces similar encodings (in the hidden layer) of all inputs patterns which have the same output. This creates a problem for prediction based on learned features since there is no way to distinguish between different input patterns. As shown in this section, making this distinction is crucial since the correct prediction of the next input pattern depends on the relevant features of the current input pattern, not just the output classification of the current pattern.

One might suspect that a problem with using a large hidden layer is that it will encode the irrelevant features in this task. To show that it does not, an experiment with using the saliency map, 30 hidden units, and N=1, was also conducted. The results are *slightly improved* over the results with 12 hidden units. The irrelevant features in this task are not encoded. Using a larger hidden layer aids in discovering a distributed representation of

Marker Position	Desired Output	Hidden Unit Activations (2 hidden units)	Difference from Average Hidden Unit activation of each class. Average RMS difference for each output.	
			avg '0' Class	avg '1' Class
	0		0.04	1.28
	0		0.03	1.25
	1		1.25	0.04
	1		1.29	0.05
	1		1.24	0.08
	0		0.02	1.27
	1		1.30	0.09
	1		1.26	0.01

Figure 3-6: Hidden Unit Activations are shown for 8 of the 16 states. Note that many are similar because of the compression in the hidden units. Only 2 hidden units are used. Note that there is less variability within classes than when 12 hidden units are used, see Figure 3-5.

patterns which can be used by the prediction mechanisms. In Section 5, the drawbacks of a large hidden layer will be discussed.

4. COMBINED DETECTION AND RECOGNITION TASKS

In some real-world vision based tasks, properties of an object, as well as the location in which the object was found, give clues to where the object will be in the next time-step. For example, in visually tracking a moving car, if a turn signal begins flashing, this indicates that the car will appear shifted from its current location towards the direction of the signal. A property of the car itself gives an indication of where it will appear in the next-time step. In an analogous manner, we can extend the cross detection task to examine traits of the cross as well as its location to determine where the cross will next appear.

The cross detection task described in the previous section is extended to a detection and discrimination task. The input size remains the same, but instead of only '+' appearing, now either a '+' or a 'x' appears in one of the 16 locations. The first objective is to determine whether there is a 'x' or '+' in the inputs. The second is to determine where the 'x' or '+' is. [Jacobs *et al.*, 1990] studied similar tasks, termed "what-and-where"; their main focus was automatically decomposing tasks in modular network architectures.

There are three factors which make the task described above difficult:

1. In every iteration, at least one marker appears. The form of the marker, whether it is a 'x' or '+', is determined randomly, with equal probability of 'x' or '+'. This creates a restriction on the types predictions which can be made. Although the position of the next marker can be determined, the type cannot. Therefore, for the next-time step's prediction, a reduced resolution image is used (4x4 instead of 20x20). This resolution effectively removes the types of the markers so that only the position of the markers are discernible. This hints at a limitation of pixel-based filtering methods (*i.e.* what to do when the activations of individual inputs cannot be predicted); these limitations are discussed in Chapters 7 and 8.
2. Like the problem explored in section 1, noise markers can appear. These noise markers can appear as either '+' or 'x', in any location. Therefore, as in the previous experiments, the sequence of positions must be considered.
3. The position of the marker at time $t+1$ is determined by the position of the marker at time t , and the type of marker at time t . Therefore, given a marker at location \mathbf{Z} at time t , the location of the marker at time $t+1$ will depend on whether the marker at time t was a '+' or a 'x' and \mathbf{Z} .

The transition matrices for the markers are shown in Figure 3-7a. An example of how to interpret these transition matrices is shown in Figure 3-7b & c.

A. Transition Matrices

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Position
Numbering
(for reference
to figure B).

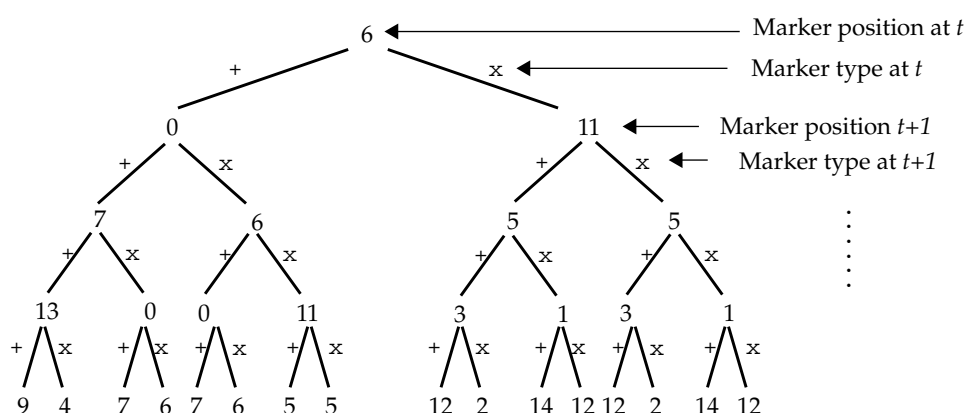
11	4	8	1
6	0	10	12
9	14	3	15
2	13	5	7

Transition
Rules for
'+'
(position order:
5,3,12,10,1...)

10	14	7	6
4	13	11	9
5	1	8	12
15	3	0	2

Transition
Rules for
'x'
(positions ord:
14,9,15,13,4...)

B. Current marker's position is dependant on last marker's position and last marker's type.



C. Sample Input/Output

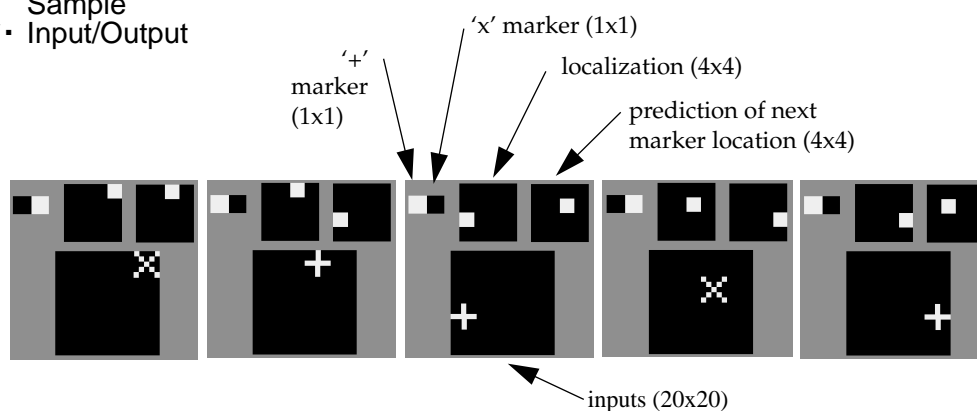


Figure 3-7: A. Transition matrices for '+' and 'x'. B. Interpretation of transition matrices. Start out in position 6. If the current marker is a '+', the next marker will appear in position 0. If the current marker is a 'x', the next marker will appear in location 11. C. Sample input and target outputs for the combined detection and localization task, given no noise in the inputs. Five consecutive time steps shown.

The remainder of this section describes the three experiments which were conducted with the framework described above. The experiments attempt to identify what type of information is needed in the hidden layer in order to perform the prediction task well. In this experiment, two pieces of information were required for the prediction: the type of the current marker and the location of the current marker. Following the methods described to this point, this information must be stored in the neural network's hidden layer, since the predictions are based upon their contents. The information is encoded in the hidden layer by training the network to perform the task.

In the next section, both the location of the current marker and the type of the current marker are used as training (supervised) signals. In the subsequent sections, only one of these two signals is used for training. The network architecture in these experiments uses a hidden layer. However, the hidden units are not necessary. For example, one of the receptive fields which could develop to solve the task is shown in Figure 3-8. Using hidden layers serve two purposes: to reduce the number of connections without a hidden layer, and to demonstrate that the content of the hidden layers dictates the accuracy of the predictions.

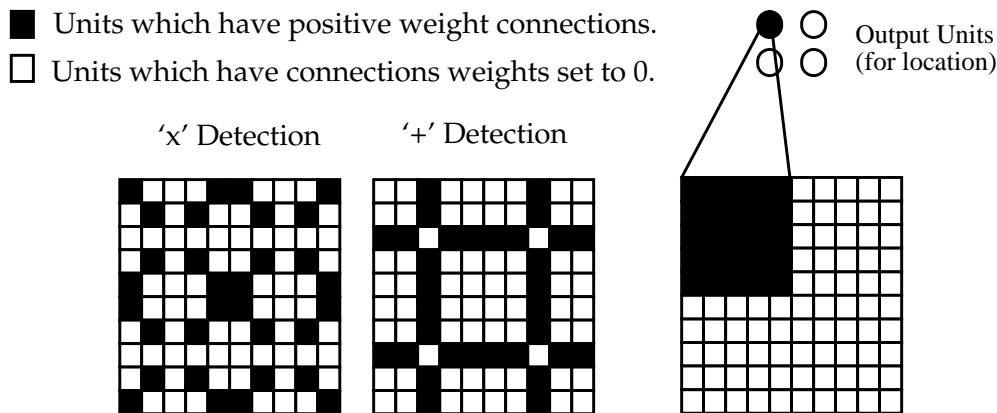


Figure 3-8: Left & Middle: Weights needed to solve the identification task without hidden units (shown for only 10x10 inputs). Left: units in the input layer that need to have positive connections for 'x' detection. Middle: units in the input layer that need to have positive connections for '+' detection. Note that since the center unit of each 5x5 square is the same for 'x' and '+' detection, it can be ignored for both. Right: Weights needed to solve the localization task. 5x5 inputs corresponding to output unit need high positive activation, all other input units should have connection weights of 0.

4.1 Two Supervised Signals: Location and Identification

The network architecture is shown in Figure 3-9. There are two hidden unit groups (#2 and #3) used for both of the tasks (localization and discrimination), which are retinally connected to the input layer as shown in Figure 3-10. With retinal connections, the hidden units are connected to only a small set of the input units (in standard fully-connected architectures, the hidden units are connected to all of the inputs). It should be noted that

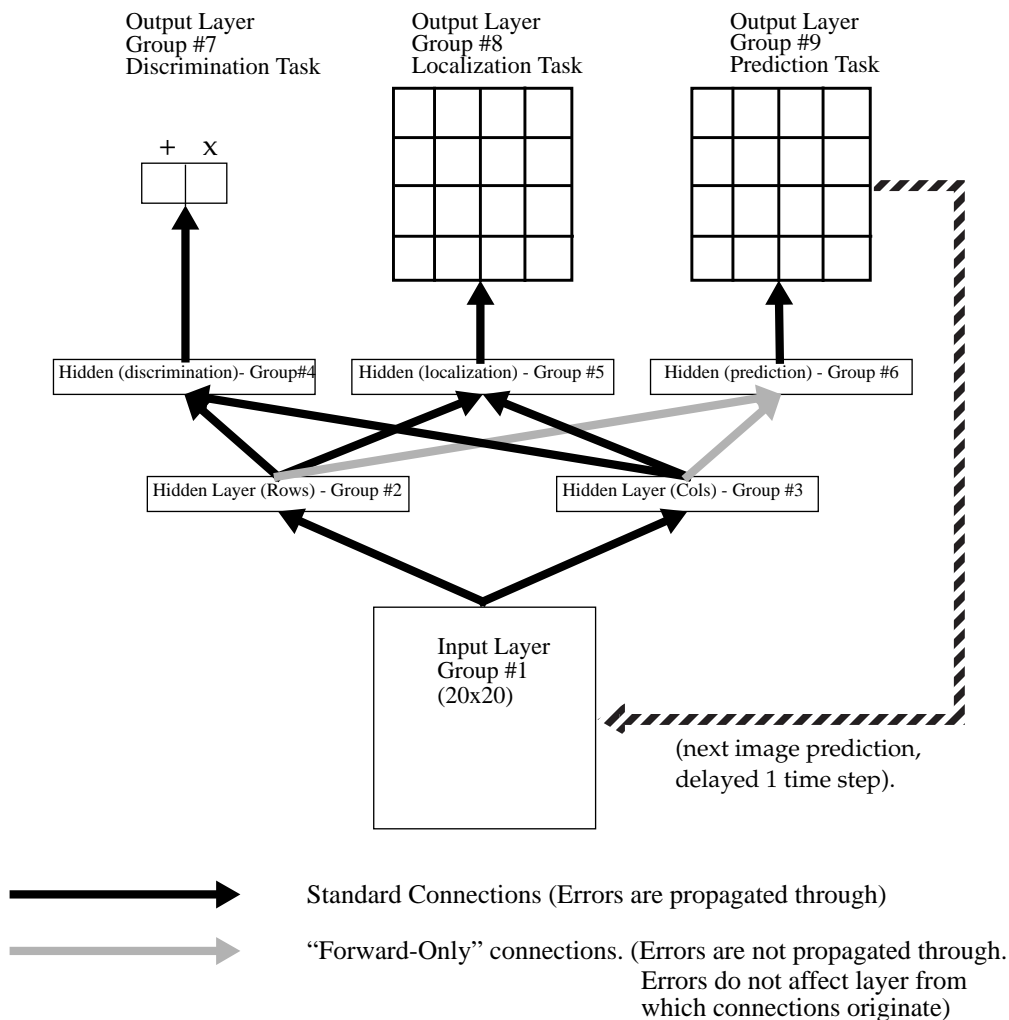


Figure 3-9: Network architecture used for discrimination and detection tasks, shown with supervised signals from both tasks (discrimination & localization). Note that because of the “forward-only” connections, error signals from the prediction task do not affect the encodings formed in groups #2 & #3. Connections between groups 1-2 and 1-3 are described in more detail in Figure 3-10.

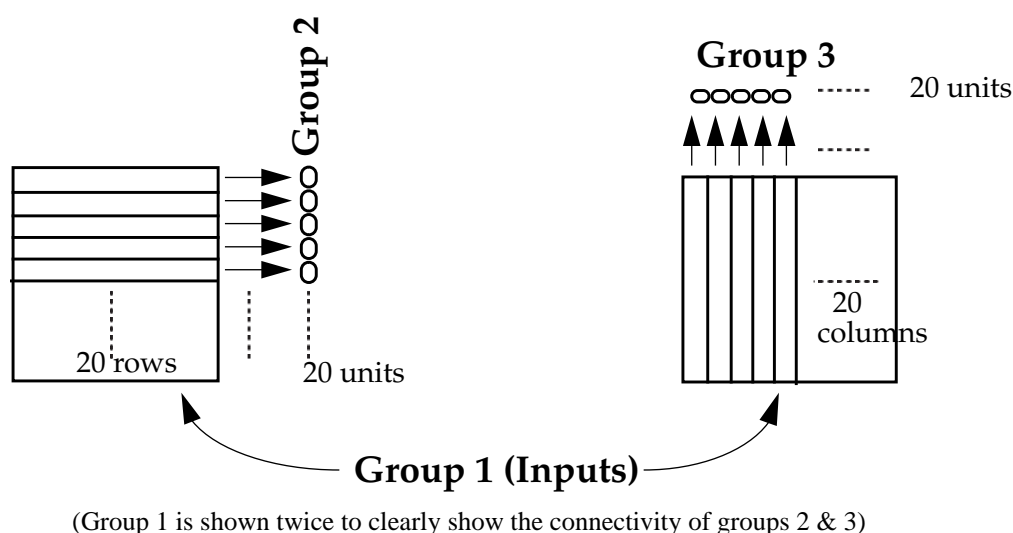


Figure 3-10: There are retinal connections between groups 1-2 and 1-3. Each of the hidden units in groups 2 & 3 are only connected to a small subset of the inputs in group 1. Each unit in group 2 is connected to 20 units in group 1, along a single row. Each unit in group 3 is connected to 20 units in group 1, along a single column.

for this task, in which location and identification are supervised signals, the retinal connections are not needed; they are useful in the experiments described in section 4.2, and will be described in detail there. In this section, they only serve to reduce the number of connections compared to a fully connected network. All three subtasks (localization, discrimination and prediction) also have a hidden unit group dedicated only to them. Note that the connections to group #9 do not affect the training in hidden unit groups #2 and #3; therefore, the learning for the localization and/or prediction task is not affected by the learning of the prediction task. Nonetheless, since the supervised output for the task comes from both group #7 (discrimination between '+' and 'x') and from group #8 (location), we would expect that enough information is contained in the hidden units to correctly perform the prediction task.

Results for this network, trained with and without noise, and with and without the saliency map, are shown in Figure 3-11 and in Table VIII. As can be seen in the table, there is no need for the saliency map when there is no noise, since both the discrimination and localization tasks can be completed without it. However, when noise is introduced, training with a saliency map provides a tremendous benefit over not using a saliency map. If

there is no noise in training, the introduction of noise in testing is detrimental with or without a saliency map. This is expected since the training and testing sets are not representative of each other. However, even if there is only 1-noise marker used in training, a network which employs the saliency map does well with no noise, 1 noise marker, or even 2 noise markers during testing.

Table VIII: Discrimination and Localization, Both Supervised, SSE on 1000 testing examples.

Training Set	Testing Set					
	0 Noise		1 Noise		2 Noise	
	Discrim. Error	Location Error	Discrim. Error	Location Error	Discrim. Error	Location Error
0 Noise (No Saliency)	0	0	129	247	172	376
0 Noise (Saliency)	0	0	150	44	186	143
1 Noise (No Saliency)	0	0	127	261	181	461
1 Noise (Saliency)	0	0	13	2	23	8

4.2 One Supervised Signal: Identification

As shown in the previous section, with two supervised signals, the system performs well. This is expected since the first hidden layer encodes information about the position and type of marker. More interesting scenarios occur when only one supervised signal is given. In this section, the supervised signal is the marker discrimination; the location of the marker is not given. Because of the noise markers in the input, the lack of location information makes this a difficult problem. It is not guaranteed that the hidden layer will contain enough information to accurately make predictions of the next marker's location. The network architecture used for this problem is shown in Figure 3-12.

In this task, the retinal connections between groups 1-2 and 1-3 will be important. However, the consequences of using a fully connected hidden layer with only the identification supervised signal should be considered. If the hidden layer is fully connected to the input layer, the hidden layer can contain activations which may not maintain spatial information, since solving the discrimination task is not dependent upon solving the location task. For example, the weights shown in Figure 3-8 (left and middle) could

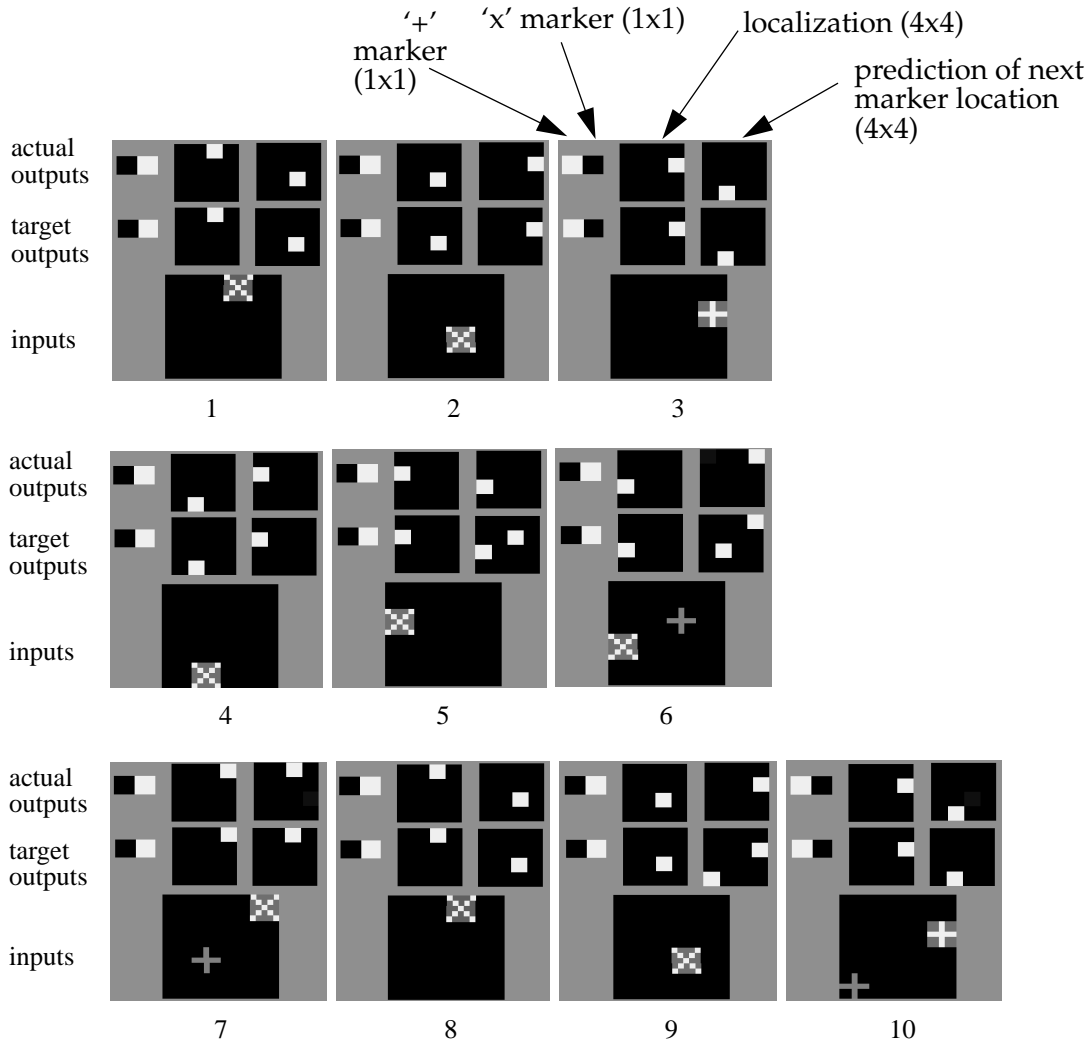


Figure 3-11: Performance on localization and discrimination tasks, with both tasks supervised. Ten frames are shown. Note that the input sometimes contains de-emphasized markers, these are the noise markers. Also note that in the prediction of the next image, the target prediction often contains a noise marker; this marker is (correctly) not predicted. If it was predicted, the focus of attention would not work. Also note that the correct marker appears “highlighted” in the inputs, this is due to the prediction only being able to determine the correct location of the marker, not the type of the marker (since this is random each step). Highlighting occurs because the pixels which do not match are scaled towards activations of 0.0 not -1.0 (the background). This effect has the potential to cause ‘ghost’ features if the prediction is incorrect. This is explained later in this chapter.

develop. With these weights, no location information is maintained, although the discrimination task is successfully completed. Since the prediction task relies on both location and discrimination information, it will not work.

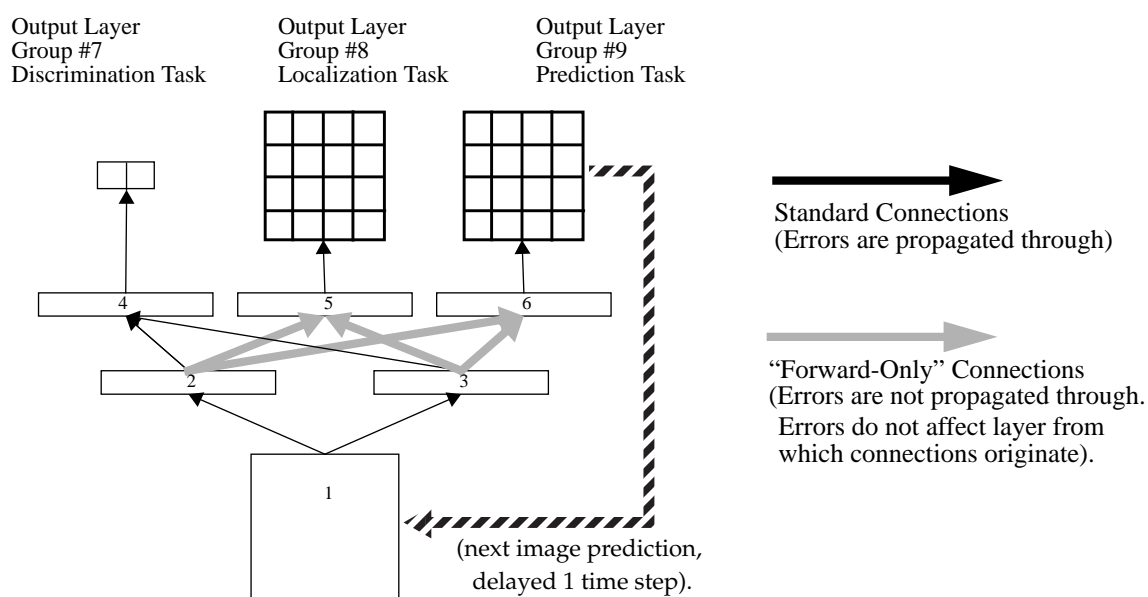
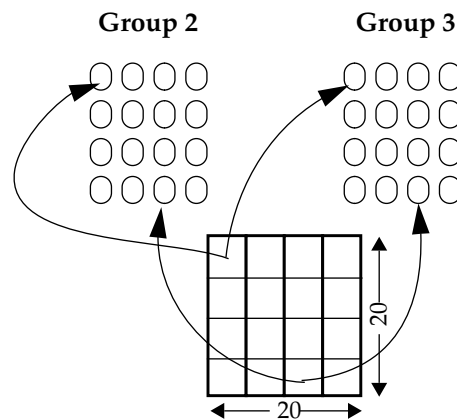


Figure 3-12: Network architecture used for discrimination and detection tasks, shown with supervised signals from only the discrimination task. See Figure 3-9 for description.

To maintain location information, even when there is no supervised location signal, the retinal connectivity is important. Since the hidden units only contain connections to a spatially restricted set of input units (as opposed to being fully connected), the activation in the hidden units tells which input units are activated. Therefore, the prediction problem can be solved. It is not critical to use the specific connectivity used in these experiments. Other connectivity architectures, such as the ones shown in Figure 3-13, can also be used.

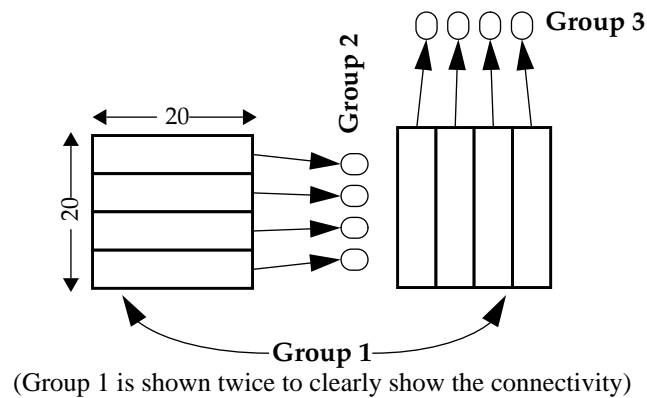
The results with complete connections are shown in Table IX, and results with the retinal connections are shown in Table X. As suggested above, the architecture which uses the retinal connections performs better than the one which does not (when a saliency map is used). However, the architecture without the retinal connections does not catastrophically fail because of the reasons mentioned in Section 3 – since the hidden layer was not a bottleneck, some unique representations are formed in the hidden layer, although they are not required for the task.



Architecture A:

5x5 retinal connections

Although it is possible to train with only group 2, empirically, training is faster with both groups 2 & 3.



Architecture B:

5x20 retinal connections &
20x5 retinal connections.

Figure 3-13: Alternate retinally connection architectures. A: Each hidden unit is connected to a 5x5 block of the input group. B: A combination of the architecture discussed in the text and Architecture A. Units in group 2 are connected to 5x20 units in group 1; units in group 3 are connected to 20x5 units in group 1.

Table IX: Discrimination & Localization, Discrimination Supervised, SSE on 1000 testing examples. Without Retinal Connections

Training Set	Testing Set					
	0 Noise		1 Noise		2 Noise	
	Discrim. Error	Location Error	Discrim. Error	Location Error	Discrim. Error	Location Error
0 Noise (No Saliency)	0	31	127	271	176	428
0 Noise (Saliency)	0	17	103	181	145	305
1 Noise (No Saliency)	0	0	129	321	177	490
1 Noise (Saliency)	0	2	21	47	67	186

Table X: Discrimination & Localization, Discrimination Supervised, SSE on 1000 testing examples. With Retinal Connections.

Training Set	Testing Set					
	0 Noise		1 Noise		2 Noise	
	Discrim. Error	Location Error	Discrim. Error	Location Error	Discrim. Error	Location Error
0 Noise (No Saliency)	0	51	130	273	170	401
0 Noise (Saliency)	0	1	124	92	160	191
1 Noise (No Saliency)	0	0	130	275	184	463
1 Noise (Saliency)	0	1	17	10	29	32

Note that the results with the retinal connections and the saliency map (Table X) are fairly close to those obtained by using two supervised signals (Table VII). This indicates that the retinal connections effectively maintain the necessary spatial information.

4.3 One Supervised Signal: Location

In the previous section, spatial location was not a supervised training signal, but it could be inferred from the activation of the hidden units. This was possible because of the retinally connected architecture. In this section, we again examine the effects of having only a single supervised signal. In these experiments, however, the supervised signal is location rather than discrimination. The network architecture is shown in Figure 3-14.

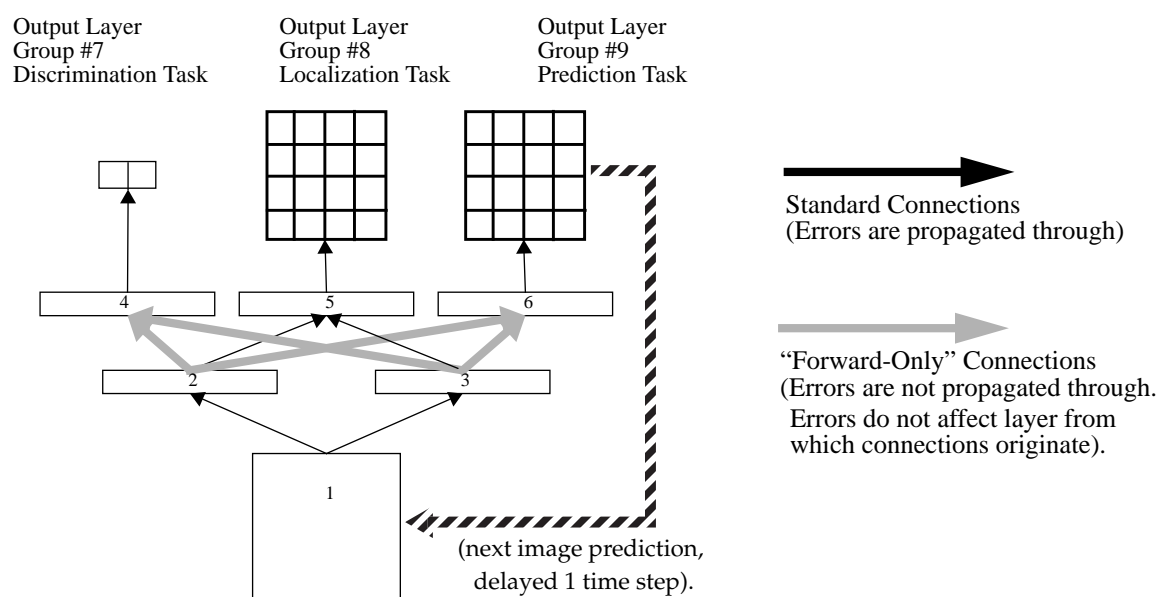


Figure 3-14: Network architecture used for discrimination and detection tasks, shown with supervised signals from only localization task. See Figure 3-9 for description.

In the experiments reported in this section, location is a supervised signal but discrimination is not. In this scenario, the prediction task is not guaranteed to be solved. The supervisory signals only ensure that the location of the marker is correct; the discrimination between shapes of the markers is not rewarded. However, both pieces of information are needed for accurate prediction. Very simple weight patterns, which do not encode type-of-marker information, can be used to detect the presence of markers. For example, see

Figure 3-8 (right). The results of the experiments with this training signal are shown in Table XI. As can be seen by the results when noise is added, the saliency map does not help performance as much as before, since accurate prediction of the location of marker cannot be made. As described in section 1.2, using a saliency map still provides an advantage over not using one. Although the exact location of the next marker cannot be determined, the next marker's location can be narrowed to only two locations (see Figure 3-15)¹. The marker will be predicted in two locations: where it would have appeared next if the current marker had been a '+' **and** to the location the marker would have appeared next if the current marker had been an 'x'.

Table XI: Discrimination & Localization, Localization Supervised, SSE on 1000 testing examples

Training Set	Testing Set					
	0 Noise		1 Noise		2 Noise	
	Discrim. Error	Location Error	Discrim. Error	Location Error	Discrim. Error	Location Error
0 Noise (No Saliency)	0	0	181	241	268	382
0 Noise (Saliency)	14	4	309	200	371	334
1 Noise (No Saliency)	120	0	273	273	278	496
1 Noise (Saliency)	266	0	307	15	353	74

Note that the discrimination errors *increase* with the use of the saliency map. Since the task of interest is localization, these errors in discrimination are acceptable. Nonetheless, it is interesting to see why this increase happens. Since the filtering methods move the activation of the inputs towards 0.0, the filtering distorts the inputs. The hidden units are *not* trained to correct for these distortions, because they have no supervised signal for the discrimination task. Thus, the discrimination error increases.

The location task improves through the use of a saliency map, but it does not perform as well as when both discrimination and localization are used as training signals. One method of improving the performance is to use the prediction of the next input image as

1. As shown in Figure 3-15, incorrect predictions sometimes lead to "ghost" features, these will be addressed in the next section.

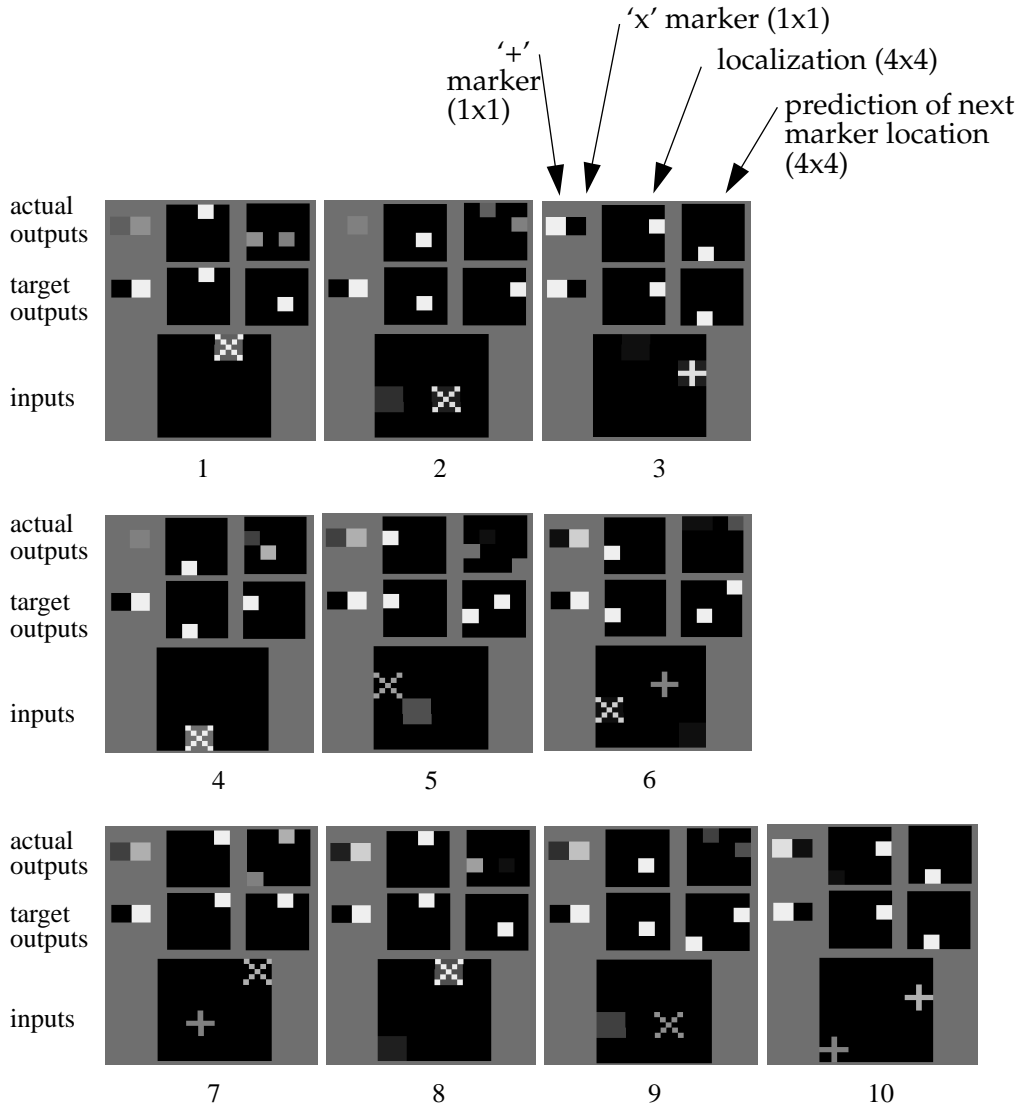


Figure 3-15: Only localization supervised. Ten frames are shown. Note that incorrect predictions are often made. This is due to not having enough information in the hidden layer to determine whether the marker is a '+' or 'x'. Also note in 5 & 9, incorrect predictions from 4 & 8 cause “ghost” markers to appear. This is where a feature is predicted, but there is none, therefore the activation value lowers. In images (1,2,4) there appears to be no actual output for the discrimination task – the output is just in a middle state, and blends with the background.

another training signal for the hidden units, in the same way as location is used. This does not require discrimination to be a supervised signal, but it does allow discrimination information to be indirectly inferred. In this synthetic experiment, this technique may work well. However, it is not a general cure. For example, in the autonomous road

following task described in Chapter 2, the network could be trained to predict all of the next input image. However, in this setup, both the prediction and the steering direction tasks will compete for “space in the network”, or for internal representations. Further, the predictions based upon this technique will not be task-specific. This conflicts with the desire for the prediction to be based upon what is important for the particular task. The task should provide information on what portions of the scene need to be attended. This will be explored in greater detail in Section 5 and Section 6.

4.4 Ghost Features Due to Incorrect Predictions

As was seen in Figure 3-15 (frames 5 & 9) incorrect predictions can sometimes lead to the appearance of features which are not in the original input image¹. Before understanding how these can be eliminated, the reason for their existence should be understood. Suppose a marker is *incorrectly* predicted (from image t) to be in some position (in image $t + 1$). This will cause the region in which the marker is predicted to be de-emphasized. The manner in which we have been de-emphasizing features is by reducing the activations of their pixels in the input layer towards 0.0. However, because the background in the input layer is set to -1 (crosses are set to +1), de-emphasizing features actually causes a region to stand-out from the background. Therefore, a “ghost” feature appears.

In these artificial domains, we knew *a priori* the background color. Therefore, rather than moving the input activations closer to 0.0, they could have been moved towards the average background color (in this case it is -1). However, in many tasks, the background color will not be as readily available; these tests were designed to show the limitations of this type of filtering in such domains. In some tasks in which the pixel’s (or region’s) average

1. Although perhaps superficial, it is interesting to note the similarity between these mistakes and the errors of expectation that humans make. Expectations often directly influence what is perceived. An interesting study to examine these types of errors was conducted as follows: with a tachistoscope, one group of subjects was told to expect to see words dealing with birds or animals, and a second group to expect words dealing with transportation or travel. The words were shown for only a 1/10 of a second. Among the words shown were combinations of letters which were not real words, but resembled real words. For example, when “*Pasrort*” was shown, the first group perceived “Parrot” while the second perceived “Passport”. Some other examples: “*Dack*” (1) “duck” (2) “deck”; “*wharl*” (1) “whale” (2) “wharf”; “*saef*” (1) seal (2) sail. This study as well as numerous similar ones are reviewed in introductory psychology text books such as [Kagan & Havemann, 1976]. Another study has shown that even in the time-span of a single experiment with a subject, the responses given on one trial effects succeeding responses, at least in absolute judgment experiments with feedback. See [Staddon *et al.*, 1980] for more details. Another interesting experiment, in which a person’s expectations and conditioning compete with the correct response is described in [Stroop, 1935] and reviewed in [Kellogg, 1995].

intensity differs in each portion of the image, perhaps it is reasonable to move towards the intensity of the pixel (or surrounding region), averaged over all samples in the training set. However, if the background color changes dramatically during testing, these filtering procedures may have difficulty. One alternative to explicitly filtering the inputs is to use the expectations directly as extra inputs. This procedure is explored in Chapter 5.

Ideally, the best method of getting rid of ghost features is not filtering on the pixel level, but rather at the object level. For example, if the majority of pixels of some object are deemphasized, then it should be removed and replaced with the “background”. Unfortunately, this type of filtering is not currently possible, since the segmentation of objects and foreground/background cannot reliably be done. In fact, it is exactly this sort of task that the selective attention mechanisms are trying to improve. This will be expanded on in Chapters 7 and 8.

4.5 Summary of Combined Detection and Recognition Tasks

The synthetic experiments presented in Section 4, which attempted to discriminate the marker type and to localize the marker, were designed to examine the effects the encodings in the hidden units have on prediction ability. By creating a prediction which depended upon two features and by controlling which of the two features is trained upon, the errors in prediction could be examined. This type of combined task also has real applications. For example, a project at CMU involves the visual detection and tracking of cars¹. Traditionally, local search heuristics are often used: If a car is present in a scene at time t , then assuming a fast enough sampling rate, the search for the presence of the same car at time $t+1$ can be limited to nearby locations to where it was found at time t . The contribution of the techniques presented here is to show how attributes of the object of interest and the location of the object of interest can be used to narrow the search for the next position of the object. In the case of a car, we know that because of the physical constraints, it can either move backwards or forwards, but it cannot move sideways. The relevant attribute of the car (the direction in which it is facing), as well as the current location of the car, dictate where the search should next be focused.

1. Project is being investigated with Henry Rowley & Takeo Kanade in the Robotics Institute, Carnegie Mellon University.

5. WHY NOT USE PRINCIPAL COMPONENTS ANALYSIS?

A final question which remains to be answered is: Why not use auto-encoding methods, such as principal components analysis (PCA) or auto-encoder networks instead of the filtering described in this chapter. This question arises most clearly in the cases in which attention is used to highlight the unexpected (for example in anomaly detection, as explored in Chapter 5). To answer this question, we take a step back from using the prediction mechanisms and examine when neural networks have an advantage over auto-encoding methods.

PCA is not task-specific. Therefore, depending on the problem, it may be very inefficient at capturing the correct features for solving the problem. For example, suppose the input retina is composed of a 20x20 pixel grid, but, the desired output is only a function of 5 of the 400 inputs. PCA will not immediately capture this since the majority of the representation will be spent on the other 395 inputs, if they also have varying inputs. *It has no notion of which inputs are important for the task.*

Recently, many studies have been released which use either auto-encoding networks or PCA to detect anomalies. The technique often used is based on [Japkowicz *et al.*, 1995]. In this technique, samples of the process to be monitored are taken while there are no errors and are used to train an autoencoding network or PCA-based system. During simulation, the input to be classified is projected onto the principal components and back onto the image (or onto the output layer of the auto-encoder network). If the difference between the reconstruction of the input and the actual input is above a set threshold, the input is considered to be from a different class than what the system was trained on, thereby signaling an anomaly. These systems have the problems of non-task-specificity. For example, an error which can only be detected by examining a few of the inputs may not be captured by these methods, depending on the number of principal components used and the variation of the other data in the inputs.

A simple task to illustrate this problem is given below. Imagine a 20x20 input in which only the first 5 inputs in the top row are important. The rest of the inputs encode structured, though irrelevant, information. In this case, the other inputs contain a horizontal

line somewhere in 1 of the other 19 rows. The task is to tell which of the first 5 inputs is activated. Samples of the input and target output are shown in Figure 3-16.

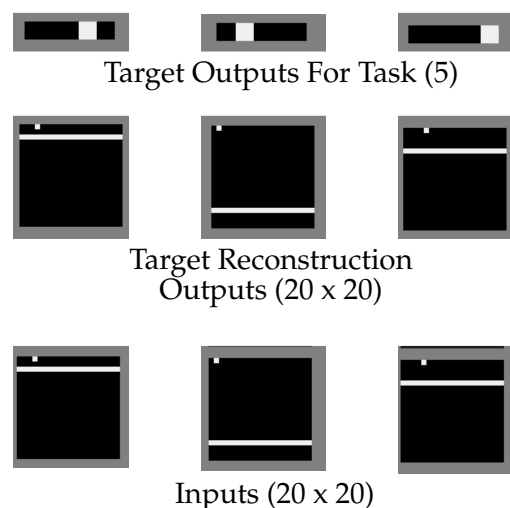


Figure 3-16: Sample I/O. Note that only the first five inputs in the top row are important for solving the task. The bar across the other rows is irrelevant.

For these tests, standard PCA is not used. Rather, its neural network analog, auto-encoding networks are used instead (see [Baldi & Hornik, 1989][Kramer, 1991] for more details). The networks for these tests are trained in three configurations. Configuration A receives error signals from the task as well as the reconstruction. Configuration B receives error signals only from the reconstruction, thereby simulating actions similar to that of principal components through autoencoding. The attempt to predict the output is based solely upon the internal representations in the hidden layer. Configuration C receives error signals only from the task itself. These configurations are shown in Figure 3-17.

The errors for these configurations are measured with three metrics. The metric of interest is determining the correct output. The errors for the correct output are shown in Figure 3-18a for each configuration as a function of the number of hidden units that are used. The second error metric that is used is the reconstruction of the 5 important units. As is expected, the ability to reconstruct these 5 units directly corresponds with the ability to do well on the task (see Figure 3-18b). Finally, the errors are also shown for the recon-

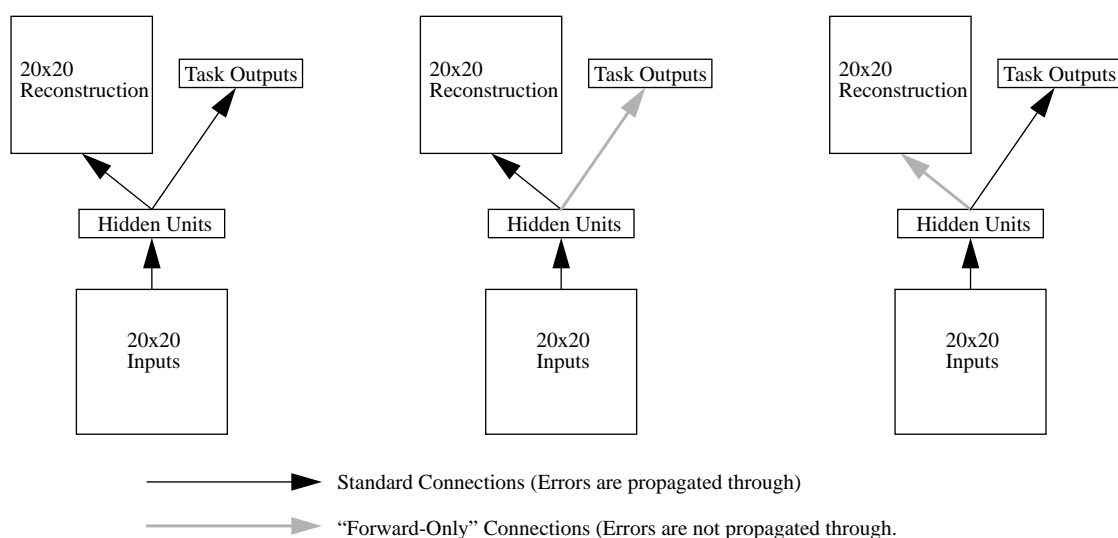


Figure 3-17: Configurations A (left), B (middle) & C (right). The configurations differ in the supervisory signal. Note that configuration B is trained similar to a standard autoencoding network.

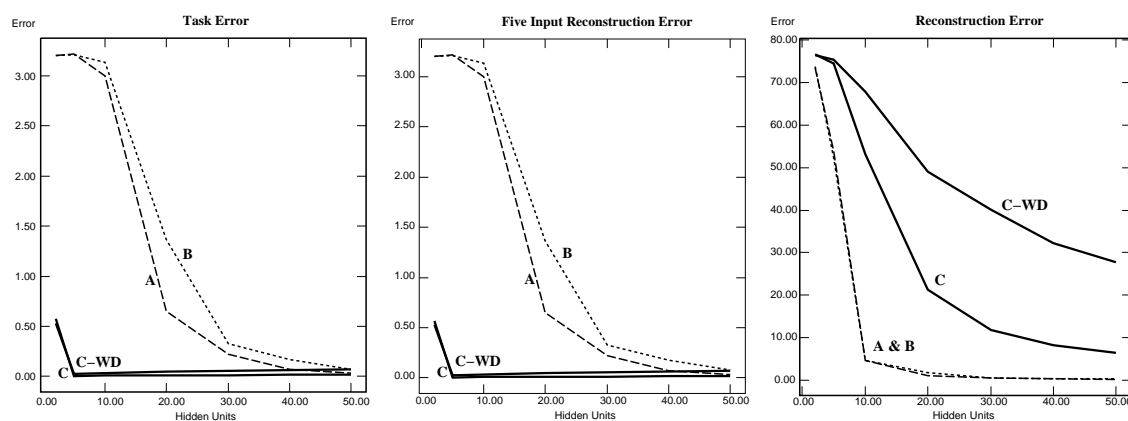


Figure 3-18: Errors shown for the three configurations for various numbers of hidden units. Each point represents the average of 10 runs (per number of hidden units, per configuration), started with random initial weights. Error is average SSE for each sample in the testing set. Also shown is configuration C with a modest weight decay (C-WD) used during training.

struction of the entire input. As expected, the autoencoder nets perform the best here. This indicates that the hidden units encode information which is irrelevant to the task of interest (see Figure 3-18c). Figure 3-19 shows the results for three examples.

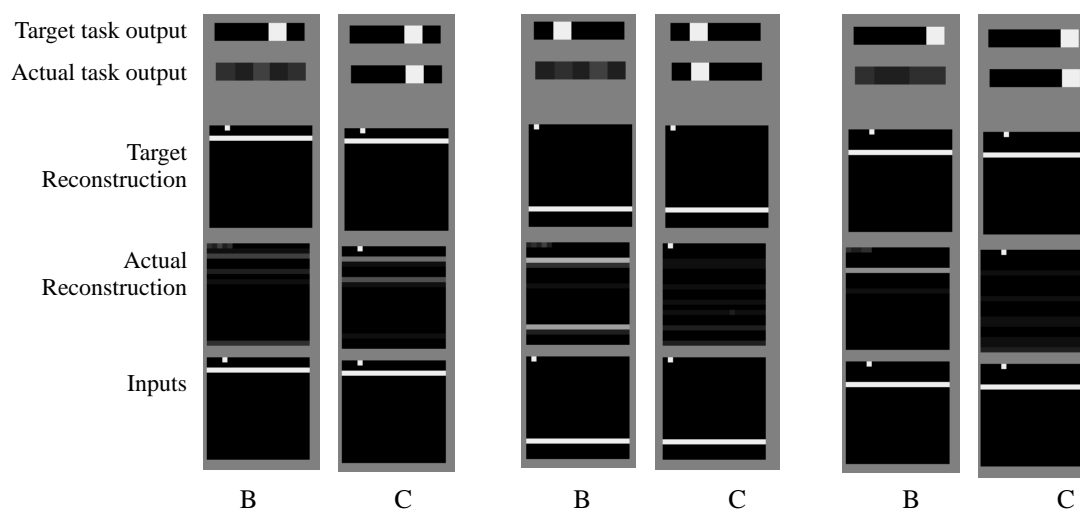


Figure 3-19: Sample performance for Configuration B and C on three examples. Note that configuration C can solve the task, but Configuration B performs better on the reconstruction. 5 hidden units were used for both networks.

5.1 Discussion

There are several interesting things to note about the results presented in Section 5. First, and foremost, is the need for task-specificity. In problems in which only a small fraction of the inputs are necessary for solving a task, PCA or auto-encoders may not efficiently encode the necessary information, and may instead require many hidden units. Eventually, however, these methods will encode the correct information, because with enough representational power, these methods will encode everything!

Second, although Configuration C is used to capture only the task-specific information, other information, such as the bar which appears in rows 1-19, can also pass through. This is especially true when there are a large number of hidden units. This is possible because encoding the information for the task (the activation of the first 5 units) does not require the use of all the hidden units. Since the weights to the other units rarely are set to 0.0, their activation provides information about the activation of other inputs. In the task examined here, this does not hurt performance. However, consider the types of domains which we have been looking at so far in this thesis. Suppose the hidden layer was used

for prediction. If information about non-relevant features enters the hidden layer, then it is possible to make predictions about the features in the next time-step, which will hinder our ability to eliminate irrelevant features in the next time-step. However, if a form of weight-decay is used in training (this can be viewed as form of regularization [Reed & Marks, 1995]), the irrelevant information can also be suppressed, by pushing all the unnecessary weights (which were not trained to reduce the supervised signal's error) to 0.0. This is shown in Figure 3-18, in which the methods with weight decay have a high reconstruction error while maintaining a low task error.

With relation to what has been shown previously in this chapter (Section 4.2), having too few hidden units can be bad, because they might not contain all the information required for prediction. On the other hand, too many hidden units will model irrelevant features, as shown in the previous experiment. There are several ways to achieve a compromise: picking an intermediate number of hidden units, changing the forms of the hidden units (retinal connections instead of full connections – as were used in Section 4.2) to force them to hold the correct information, or by using weight decay with a larger number of hidden units. Experiments have shown that each of these works well in practice.

Third, it should be noticed that the Configuration A, which was trained with both supervised signals, did not perform better than Configuration B. This is because of the low ratio of the number of relevant to irrelevant inputs. Most of the network's effort is guided towards solving the reconstruction problem, since the error from the reconstruction swamps the error from the task.

5.2 Ramifications for Anomaly Detection

[Japkowicz *et al.*, 1995] [Petsche *et al.*, 1996] have suggested that PCA, or autoencoding networks, are a good way to find anomalous inputs by analyzing the magnitude of the difference between actual and reconstructed inputs. An autoencoder is trained on positive data. When the autoencoder is presented with negative data, the autoencoder will not be able to reconstruct it accurately. This technique is only useful when anomalies can be detected by analyzing large portions of the input. In the task described in Section 5,

only 5 of the 400 inputs are relevant. As shown in Figure 3-18, PCA does not capture information about these inputs until many components are considered. If we imposed an anomaly/not-anomaly classification on patterns based on the activation of these 5 inputs, the number of components required would make PCA inefficient. However, a neural network would be able to capture these with few hidden units, as was shown in Figure 3-18.

One of the advantages of the [Japkowicz *et al.*, 1995][Petsche *et al.*, 1996] systems is that they can be trained with mainly positive examples, with just a few negative examples to set difference thresholds. The system employed in this section must have examples of both positive and negative inputs. This reflects the difference between a *novelty detection* approach and a *classification* approach.

The task examined in this section was not temporal, so expectations were not needed. However, expectations can be used for anomaly detection in temporal domains by indicating the regions in which the expected input does not match the actual input. This will be explored in Chapter 5, in the domain of fault detection in the plasma-etch step of semiconductor wafer fabrication.

6. WHERE'S THE CATCH?

Unfortunately, one can construct an example in which methods presented in this chapter will perform poorly. Assume that a network has several real-valued inputs which are functions of time, two of which are $f(t)$ and $g(t)$, while the other inputs contain noise. The target output of the network is $(f(t) + g(t))$. Although an ANN can solve this problem easily, a hidden layer (if one is used) may not encode the values of the inputs, but rather just the final sum to be output. Since information will be lost from the input to the hidden layer, the next inputs cannot be predicted accurately (since they are based on the contents of the hidden layer). Therefore, the important inputs may be deemed as not-salient. In cases in which the hidden layer does not contain enough information for accurate prediction, the predictions can be used as another supervised training signal (as a separate task, similar to the ideas in Multitask learning [Caruana, 1993], described in Chapter 7, or as used in Section 5 - Configuration A). Although this will not make the prediction task-specific, it can provide a method to increase prediction accuracy. This increased accuracy, coupled with the feedback of the attention mechanisms, provides a means to remove noise from the inputs.

The proposed solution described above is, unfortunately, not foolproof either. Imagine three time-dependant inputs to a network $f(t)$, $g(t)$, and $h(t)$. The target output of the network is still $(f(t) + g(t))$. However, in this example, if the prediction is used as a supervised training signal, as suggested in the previous paragraph, then $h(t)$ will also be a trained signal, and may also be predicted accurately. The first problem with this is that representing this input in the hidden layer will compete with representing the important material for the main task. As was seen in Section 5 (Configuration A), this error signal may swamp the real error signal. A second problem is that since the $h(t)$ prediction can be done accurately and the mechanisms described in this chapter and in Chapter 2 are used, it will mistakenly be assumed to be a relevant input in the next time step. We would like to use "forward-only" connections between the prediction and hidden layers because of the task-specificity they provide, since $h(t)$ would therefore not be encoded. However, we are unable to use the "forward-only" connections due to the loss of information in the hidden layer.

Aid in addressing the above problem comes from an unexpected place: the distributed nature of neural network encoding. The idea is to use a large hidden layer to solve only the particular task¹. Then, as was seen in Section 3, the hidden units encode portions of the solution in a distributed manner. The information is transformed to an intermediate form in the hidden layer, rather than being lost. With a large hidden layer, it is possible to also encode extraneous information since there is extra capacity. However, as shown in Section 5, weight decay can reduce the extraneous information encoded.

Although synthetic tasks can be constructed to display some of the above problems, they have not appeared in the “real-world” applications explored in this thesis, such lane-marker tracking (Chapter 2), hand-tracking (Chapter 4), or anomaly detection in the plasma-etch step of wafer fabrication (Chapter 5). One reason that these problems do not arise in these non-trivial tasks is that the network creates features in the hidden units (including complex/hierarchical features in multiple hidden layer networks), which can then be transformed into the desired outputs. The large hidden layers used, the complexity of the task, and the training algorithms, discourage solving the task in only the input-to-hidden layer connections. Only the relevant information components are encoded in the hidden layer. From these components, the task is solved and the predictions are made.

7. CHAPTER SUMMARY

This chapter has shown that the contents of the hidden layer play an important role in determining the success of the prediction based selective attention methods described in Chapter 2 and this chapter. Attention is focused based upon the differences between the predicted and actual input images. The prediction of the next images is based upon the contents of the hidden layer. Therefore, the hidden layer must be able to capture the information necessary to make accurate predictions of the next input scene. Using too small a hidden layer often hinders performance.

It was shown that in many cases, the hidden units develop enough information for pre-

1. Problems of overfitting using a large NN can be avoided by using early stopping and weight decay.

diction just by solving the original task. Allowing a large hidden layer, such that it is not a severe bottleneck, makes it possible to preserve information necessary for prediction which may otherwise be lost. It was also shown that spatial information can be preserved, even without an explicit training signal, by the use of retinal connectivity architecture. Methods for reducing the extraneous information encoded in the hidden layers were also discussed.

7.1 Future Experiments

In the experiments presented in this study, two assumptions are made. The first assumption is that the transitions are deterministic. The second assumption is that the transitions are Markovian. In real world tasks, these assumptions are easily broken. In the autonomous road following task described in Chapter 2, and the hand-tracking task which will be described in the Chapter 4, the transitions are not deterministic. Nonetheless, given enough training data, the saliency map should be able to learn the transition probabilities for its prediction of the next input. For example, if a cross in position **X** leads to a cross in position **Y** with a 90% chance and a cross in position **Z** with a 10% chance, the saliency map will be able to come up with respective activations in position **Y** and **Z**, given input **X**. Nonetheless, there is still difficulty when the real cross appears in either position **Y** or **Z**, and a noise cross appears in the other position. In this case, of course, it will not be clear which is the correct cross. A second difficulty arises when the probability of a cross appearing in **Y** and **Z** is divided equally (50%). Since the activation of the expectation in both of these positions will be less than a full expectation ($< +1.0$), the expectations will never match the actual inputs. Therefore, the form of filtering described to this point will not be sufficient. Alternate strategies for using expectations are described in Chapters 4 and 5.

The autonomous road following and hand-tracking tasks also serve as examples for tasks which are not Markovian. For example, in the road-following task, if the difference in road position between the input at time t and the input at time $t+1$ is large, it is likely that the difference in road position between input $t+1$ and $t+2$ will also be large. Such behaviors might arise because of the physical constraints on the velocity of the vehicle.

Although it is not explored in this thesis, the methods explored here can incorporate multiple previous time steps. By using recurrent connections, or networks which use the inputs and/or the outputs of the previous few patterns, state can be maintained. Many of these methods should be easy to integrate with the selective attention methods described here and in later chapters.

CHAPTER 4

UTILIZING DOMAIN-SPECIFIC KNOWLEDGE FOR CREATING EXPECTATIONS

Spatial selective attention has been implemented with an expectation-based saliency map. To this point, the expectations have been learned simultaneously with the task to be solved. In this chapter, we consider the problem of incorporating a priori available domain-specific knowledge. For example, in the task of tracking a continuously moving object, the physics of the real-world constrain where we should expect the object to be in the next time step, given its current location. In this chapter, the saliency map is used as a general mechanism through which any such a priori available domain knowledge can be used without requiring the system to learn it. The sample task used in this chapter is visual hand-tracking in a potentially confusing sequence of images.

1. INTRODUCTION

In many vision tasks, only a small fraction of the inputs are important at a particular time. Consider object tracking in a cluttered scene, or in a scene containing many similar objects. For example, recall the synthetic tasks described in Chapter 3. In these tasks, the extraneous information in the input signal could easily be confused with the important features.

For the autonomous navigation task presented in Chapter 2, a method for finding and focusing attention on only the task-specific relevant inputs was presented. Based on the current inputs, a task-specific expectation of the next time step's inputs is computed. Processing is focused on portions of the input which match the computed expectation. This has the ability to remove many types of noise and spurious features. One characteristic which made the saliency map method well suited for general road-following tasks is the difficulty in *a priori* determining the important features on which the network should be focusing. Through training, the system was able to *automatically* determine the important features/portions of the scene to concentrate upon. However, in other problems, such as object tracking, domain-specific knowledge may be useful in limiting the portions of the input which need to be processed. In this chapter, we take a "step back" from learning the expectation of the future input. Instead, we examine how domain-specific knowledge (potentially rule-based) of future inputs can be integrated into a neural based system without requiring the system to discover this knowledge on its own.

In this chapter, vision-based hand tracking is explored. Although there have previously been fairly sophisticated models used for hand tracking [Rehg & Kanade, 1995][Cui & Weng, 1996][Bobick & Wilson, 1995], the model used here is very simple. The goal of this chapter is not to create a new model for hand movements, but rather to demonstrate (1) that previously developed models can be incorporated into the system without requiring the system to relearn the model, and (2) to show that the expectation-based methods developed in this thesis encompass many forms of filtering and tracking that have been used elsewhere. In the future, if this system is to be extended for use in vision-based user-interfaces, more complex hand-tracking models can easily replace the simple ones used here.

In the next section, vision-based hand-tracking is introduced. The hand-tracking system developed by [Nowlan & Platt, 1994] is also discussed. Section 3 extends the ideas in Chapter 2, to show how to use the saliency map when there is domain-specific knowledge. Methods for directly integrating domain knowledge into the hand-tracking system are presented. Section 4 presents empirical results with the hand-tracker. Finally, Section 5 concludes, and presents directions for future research.

2. VISION-BASED HAND TRACKING

One of the first steps in building a non-intrusive vision based hand-gesture recognition system is creating a method to locate and track a user's hand. However, this problem can be complicated because distracting features may appear in the input image. For example, in office environments, there are often other people or distracting motion in an input scene. The ability to work in the presence of this noise is crucial.

In 1994, Nowlan & Platt presented a very accurate neural network based hand-tracker [Nowlan & Platt, 1994]. However, the scenes which they considered usually consisted of only the user's hand and arm moving. Their system arbitrated between the outputs of two networks to determine the position of the hand. As input, the first network used an intensity image and the second used a difference image (difference between the current and previous inputs). The intensity image network was subject to errors because of complex backgrounds. The difference image network, which was able to achieve a lower error rate than the intensity image, still had problems because other objects were moving in the scene (for example, the person's forearm), and this could confuse the system. Second, when the hand stopped (as when reversing direction), the difference image lost the hand.

The system described in this chapter is robust to many of these errors. Because the system is able to focus attention on small portions of the scene, errors caused by complex backgrounds are reduced. To address errors of losing a motionless hand, which are due to the standard difference image pre-processing, the calculation of the difference image is modified. When a difference image is calculated for time t , instead of subtracting it from

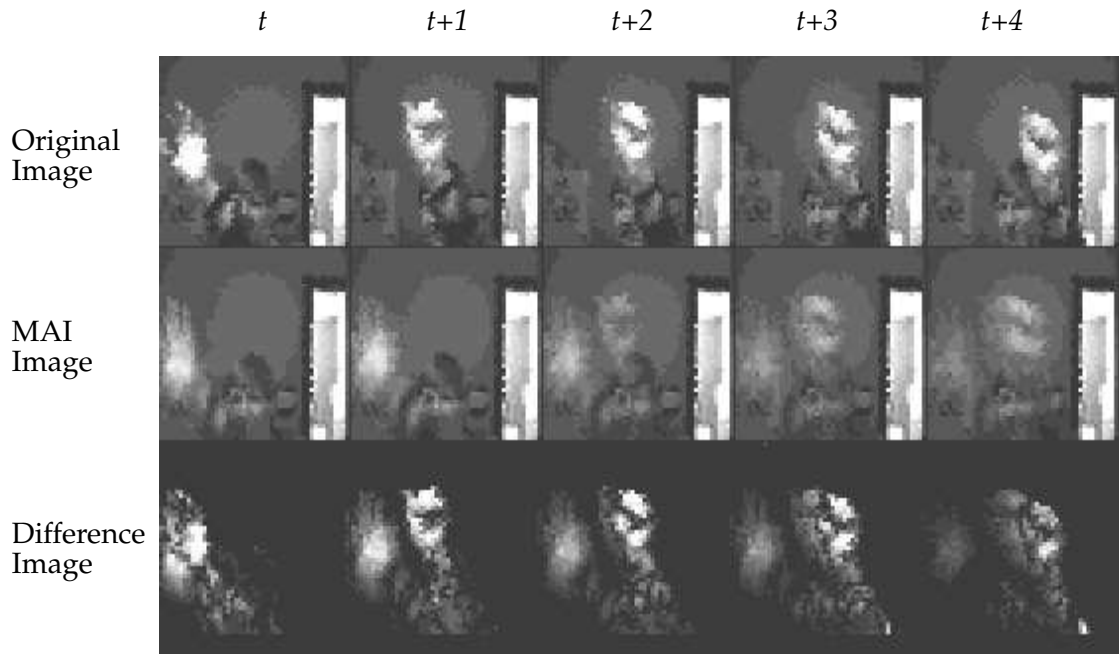


Figure 4-1: Sequence of 5 images shown. Top: Original Image. Middle: MAI. Bottom: Difference Image. The thick white bar which appears on the right side of each image is a brightly lit patch in the background.

image $t-1$ (as is often done), it is subtracted from a moving average image (MAI) of the previous few frames. The MAI is a slowly changing background image against which a moving hand or other moving objects stand out. Examples of the MAI and the difference images which are computed based upon the MAI are given in Figure 4-1.

The network used for this task is retinally connected, as described in Chapter 3:Section 4.2, and uses weight sharing to promote translation invariance of feature detection (see Figure 4-2). Weight sharing is a commonly employed technique when similar features are to be detected everywhere in the input image. In this task, it is used because a hand looks the same in all locations of the image. Therefore, the features which are used to detect a hand should be applied to all regions of the image. Other applications in which weight sharing is often used is in zip-code handwriting recognition tasks [Le Cun *et al.*, 1989], as well as neural-network based speech recognition systems [Waibel *et al.*, 1989]. The retinally connected network architecture is similar to the ones used in Chapter 3 for the synthetic experiments which attempted to simultaneously detect and discriminate between

two different markers in the input. However, the networks used in the experiments described in Chapter 3 did not use weight sharing.

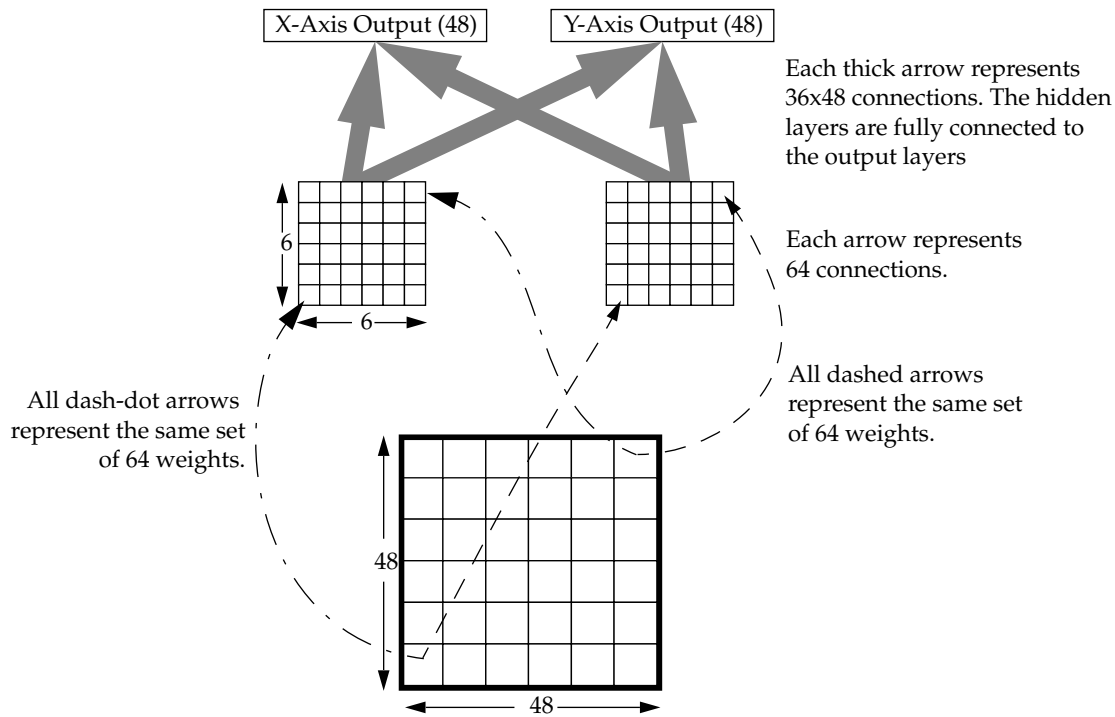


Figure 4-2: Network Architecture. There are two output layers, one representing the position of the hand in the X-axis, the other in the Y-axis. Since the network is retinally connected, each of the hidden units is connected to only a 8x8 patch of the input layer. There are two hidden units examining each 8x8 patch. For each hidden unit group, the weights from all of the units to each patch are the same. For example, each dash-dot line has the same set of 64 weights. Each dashed line has the same set of weights as every other dashed line. The feedback mechanisms are not shown; they will be described in the next section.

The network architecture used in this system is much smaller than the one used in the Nowlan and Platt system. The outputs for this system are 2 vectors (for the X and Y axes) of 48 units each. The target output is a gaussian of activation centered around the X and Y location of the hand. This is the same output representation, on each axis, as was used in the autonomous navigation task (in Chapter 2). More details on this output representation can be found in [Pomerleau, 1992]. An example input and output image is shown in

Figure 4-6.

To train the network, 1,580 training images were gathered from 3 subjects. The subjects were told to wave one hand while standing still. Throughout the sequence, each of the subjects opened and closed their hand many times. Each image was manually labeled with the X/Y coordinates of the hand (near the palm). In order to test the system, two additional sets of data were collected. The first set, which contained 300 images, was gathered from a fourth subject who waved one hand. In the second set, which contained 383 images, the subject waved *both hands* and moved, relative to the camera, throughout the sequence.

The second test set was designed to stress the focus of attention abilities of the system. In every frame there were at least two moving objects (both hands), while the desired output tracked only one hand. This is perhaps the most difficult type of distractions to ignore, since the network is tuned to be sensitive to hands. Therefore, even though other moving objects will make it past the difference imaging, they will not appear as confusing as other hands moving. In addition to the two hands moving in each image, because of the subject's movements, a third object, the subject's body/face, also often appeared in the difference image. As will be shown, a system which does not selectively focus processing is unable to track a specified hand reliably.

3. INCORPORATING DOMAIN SPECIFIC KNOWLEDGE

In this section, two methods of incorporating domain-specific knowledge are presented. The first is to explicitly create the expectations for the saliency map. The second is to post-process the network outputs.

3.1 Explicitly Creating Expectations

In this domain, temporal coherence is a large source of knowledge to disambiguate the hand being tracked from other hands or distracting features. Assuming that the sampling rate of the video is fast enough, the position of the hand of interest at time t is a strong

indicator of the location of the hand at time $t+1$. Methods to incorporate this knowledge into the neural network are given below.

A straightforward method of incorporating this knowledge is to use the architecture shown in the previous chapters to predict the next input frame. Because the system cannot measure motion, since only one frame is given as input, the network predicts the location of the hand to be close to where it was found in the current frame. Since the prediction is based only on the inputs from a single time step, the direction of motion is not available. Therefore, the network can only indicate a region around the current location. The radius of this region is a function of the video sampling rate and the speed of the hand in the training set. If methods similar to those in Chapter 2 are used, in which a 2D grid is used as the prediction output layer, the prediction outputs will predict the hand in many locations around the network's estimate of the hand location in the current frame. If the information from multiple frames is used, motion and position estimation will improve, and the number of locations the hand is predicted in will decrease. Unfortunately, there are two problems with this method. They are described below.

The first problem is that although the prediction of the location of the hand in the next image will be correct, the orientation, the number of fingers extended, etc., are more variable in shorter time spans, making them harder to predict. Therefore, comparing pixel intensities may be misleading. Since the filtering is done on a pixel-by-pixel basis (based upon the magnitude of the difference in the intensities of the predicted and actual next images), missing the orientation may cause the hand not to be detected in the next frame due to accumulation of errors (see Figure 4-3). There are two methods of addressing this problem. The first is to assume that the sampling rate is high enough to take care of the problem by always predicting the orientation seen in the previous frame. The second is to predict the *location of the hand in the next time-step rather than the intensities of the next image*. These location predictions can be used by passing the activations of the inputs close to the predicted location through to the neural network without modification, while attenuating the activations of the inputs that are far from the predicted location.

The second problem is that although we can successfully predict the location of the hand in the next frame, and can therefore use it to create a saliency map, none of the *a priori*

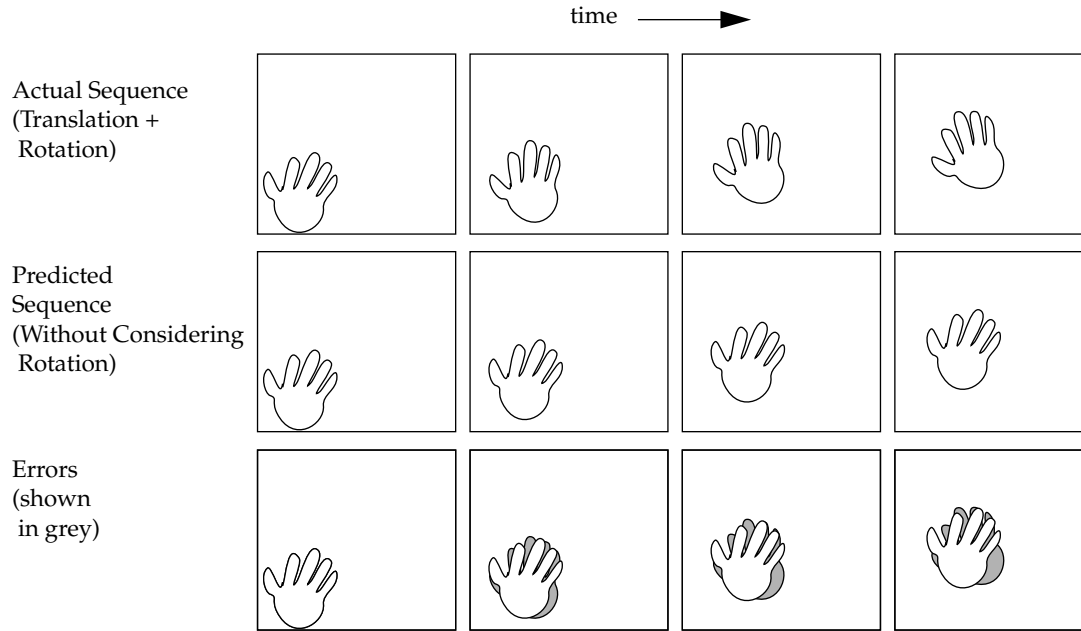


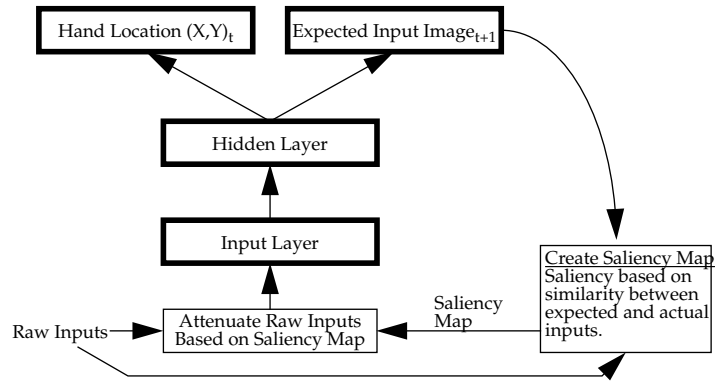
Figure 4-3: Problems with pixel-by-pixel filtering when orientation is not considered. Worst-case scenario shown, when orientation is not considered at all.

available information about the task is used. The network has to *learn* the possible transitions of a hand from one frame to the next. The purpose of this chapter is to show that these transitions do not have to be learned, and can be encoded directly into the network. Put simply, we want to tell the network to search for the hand in the next image close to where the hand was found in the current image, instead of making it learn this rule. This information can easily be conveyed by *explicitly modifying* the saliency map.

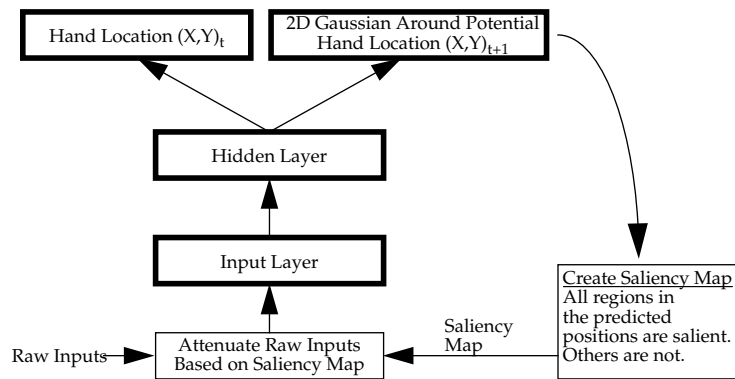
3.1.1. Network Architectures

In the filtering methods described in Chapter 2, the difference between the actual and expected inputs was the basis of the saliency map. When the next-inputs became available, they were emphasized if they correlated with what was expected (see Figure 4-4 (Top)). However, in the hand-tracking domain, to predict the next time step's input image accurately, we must be able to predict the orientation of the hand, which, as described earlier, is difficult. We can avoid this problem by training the network to predict just the location of the hand rather than the entire next image. For example, we can

The network architecture From Chapter 2 - predictions about the next input image are made.



A network architecture to make predictions about the hand's next location (not used).



Predictions are not made by the network. They are created from the domain knowledge.

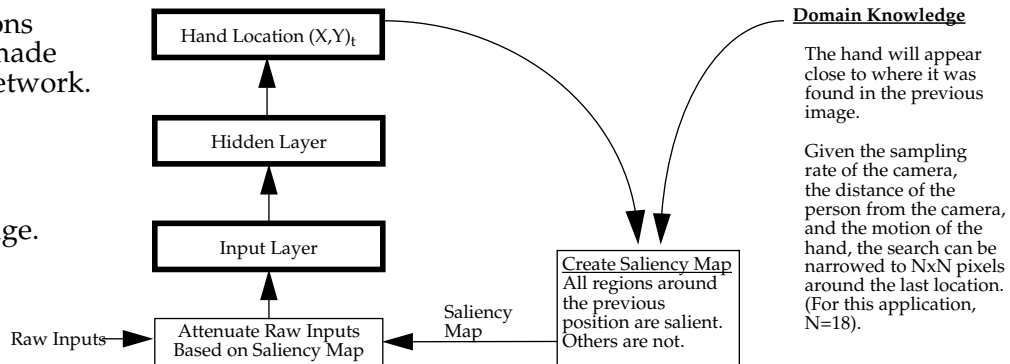


Figure 4-4: (Top) Model from Chapter 2. (Middle) Adapting the Model from Chapter 2 to make predictions - Note that this model is *not* used. (Bottom) Adapting the Model from Chapter 2 to incorporate domain specific knowledge about the predictions - *This model is used in this Chapter.*

ask the network to create a 2D gaussian around the area in which the hand should appear in the next image. A network architecture to do this is shown in Figure 4-4 (Middle). We can then use these predictions as a filter for the next time step's inputs. Only the locations in which the hand is predicted pass through the filter unaltered; all other locations are attenuated.

The architecture in Figure 4-4 (Middle) alleviates the problem of having to predict the orientation of the hand. However, it does not allow us to integrate all of the information we have about the domain. The architecture in Figure 4-4 (Bottom) shows how *a priori* information about the predictions can be incorporated. The current location of the hand and domain knowledge are used to explicitly create the saliency map. Given the sampling rate of the camera, the average distance of subjects from the camera, and the average velocity of the hand, we found that in the next frame we should search for the hand in an 18x18 region around the current location of the hand (this is approximately 14% of the entire input). In the simple model used here, the inputs which correspond to an 18x18 region around the current location of the hand are salient, while all others are not-salient. This binary saliency map is used as a pass-filter. Input activations which fall inside this window are not altered; activations outside of this window are attenuated. More complex models which use estimates of the velocity of the hand can create more sophisticated saliency maps, which may, in turn, create a more focused salient region. The methods used in this chapter are similar to the ones used in Chapter 2, except that instead of using the network's model of the next state to determine the important regions, the map is created based upon knowledge of the physical constraints of the hand motion.

3.1.2. A Note About Training

It is necessary to train the neural network with typical examples it will see during use. Therefore, even for training, the examples must be filtered as they would be when the network is being tested. Examples are created by passing the images through the pass-filter centered around the location of the hand in the previous example. In practice, however, the predictions from the previous time step will not be perfect. To account for some of this variation during training, the 18x18 window is shifted randomly, by a small amount, in both the X and Y directions before it is applied (see Figure 4-5). This some-

times cuts off portions of the hand image, as will happen in use.

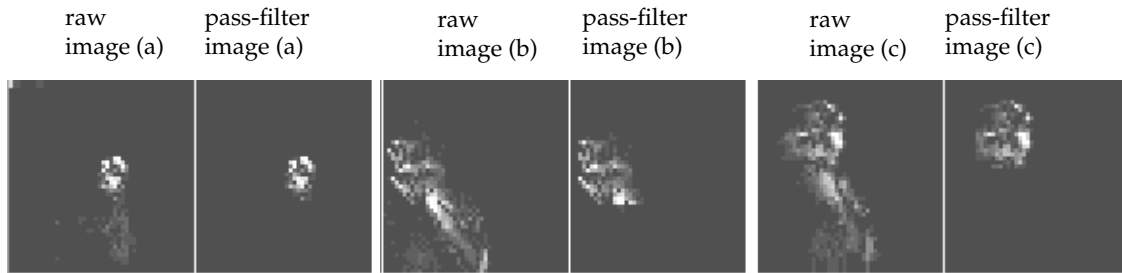


Figure 4-5: Samples used for training. Note the effect of the saliency map. Images presented in alternating order, original and after applying the saliency map filter.

3.2 Post-Processing the Output

The saliency map is not the only way of incorporating domain specific knowledge. The output of the network can also be filtered. Recall that a gaussian of activation is fit to each of the output layers to determine the network's X and Y prediction. Before this gaussian is fit, the output activation in each unit can be scaled inversely to the unit's distance away from the estimated location of the hand in the previous time step. This has the effect of reducing the probability of estimating a location which differs greatly from the previous estimation. This helps in the many cases in which a network's output layers contain several distinct gaussians corresponding to detecting both hands, as shown Figure 4-6 (left).

A second heuristic is to ignore any predictions which differ from the previous prediction by a set threshold. Prediction which differed by more than 24 units (half of a dimension of the input layer) were ignored, and the previous prediction was used instead.

When these two heuristics are combined, they improve the performance of the networks which do not use a saliency map. However, as will be shown in the next section, the saliency map out-performs these heuristics.

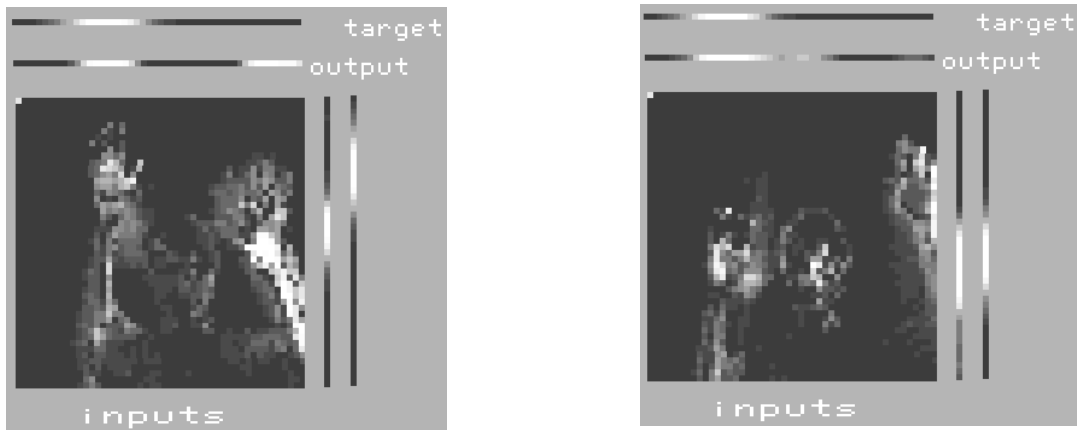


Figure 4-6: A network without a saliency map can get confused when there are multiple similar features in the difference image. (Left) Both hands found in both the **X** and **Y** outputs. (Right) Subject's right hand and face found in **X** axis output.

4. RESULTS

As mentioned before, two sets of data were used for testing. The first image sequence showed only one hand. The person in the images moved infrequently; therefore, the person's face and body were removed in most of the difference images. The second set contained images of a person moving both of his hands. In addition, the person also moved quite a bit; therefore, he is still clearly visible in many of the difference images, see Figure 4-6. Images from the hard test set, after using the MAI for difference imaging are shown in Figure 4-7 (Bottom).

The performance of networks with and without a saliency map, and with and without the heuristics, is measured as follows. The input to the network is a difference image (either with or without a saliency map applied). A gaussian is fit to the activation of the output layers. The center of the gaussians represent the system's prediction of the **X** and **Y** position of the hand. If the test includes the two heuristics mentioned above, they are applied to the outputs of the network. The system is measured as follows. If the resulting prediction is within 7 pixels (which is the size of a typical hand) of the hand being



Figure 4-7: (TOP) Original Images from easy (left 2) and hard sets (right 2).
(BOTTOM) Difference images from hard test set (top).

tracked, in both the **X** and **Y** axes, the system is considered to have correctly found that hand. (The correct spot is known as all of the test images have been manually labeled with the positions of both hands). If the system is not tracking the correct hand, the frame is considered as missed, and the correct **X** and **Y** locations replace the predicted values. It is important because the heuristics and the saliency map use the previous time step's prediction. If this correction was not made, these methods may "successfully" track the wrong hand. Replacing the incorrect values is used in all of the systems which employ the saliency map and/or the heuristics. The end result is the number of frames in which each of the hands were tracked, and a number of frames in which neither hand was tracked (possibly because of other features in the images).

The saliency map was used with different "strengths". The strength refers to the weighting of the filtered difference image, with respect to the raw difference image. For example, if the strength = 0.0, the input to the network is only the raw data. If strength = 0.66, the filtered difference image contributes twice as much as the raw difference image.

In the first test set, in which there is very little distracting content in the scenes, networks trained with and without a saliency map perform comparably, since there is no need for focus of attention. A more revealing performance comparison is on the harder test set, in which there are confusing features. This is a more realistic test set, as the subject was not

requested to stand motionless, and there are confusing features, such as other hands, in the inputs. Results are presented for tracking first the left hand and then the right hand. Both of these results are important, as an unanticipated feature in the test set was that the left hand was much easier to track than the right. This was largely due to the lighting during collection – the left hand appeared brighter.

Table I: Performance of Saliency Map and Heuristics: Number of frames in which Left hand was located (283 total images).

Saliency Map Strength	Method	Target: Track Left Hand		
		Which Hand Was Located		
		Left	Right	Neither
0.0 (No Saliency Map)	No Heur.	146	44	93
	Heur.	187	22	74
0.9	No Heur.	258	3	22
	Heur.	256	3	24

Table II: Performance of Saliency Map and Heuristics: Number of frames in which Right hand was located (283 total images).

Saliency Map Strength	Method	Target: Track Right Hand		
		Which Hand Was Located		
		Left	Right	Neither
0.0 (No Saliency Map)	No Heur.	143	47	93
	Heur.	68	147	68
0.9	No Heur.	3	255	25
	Heur.	2	257	24

The networks which use a saliency map (strength=0.9) track the correct hand in over 90% of the images. With only the heuristics (no saliency map) the performance ranges from 52-66% depending on which hand is tracked. With no heuristics, the performance ranges from 17-52%, depending on which hand is tracked. Heuristics help the performance of the networks which do not use a saliency map. Without either the heuristics or the saliency map, the network can track the left hand more accurately than the right. This is

due to the lighting conditions. Using the saliency map, the network performs equally well on both hands. Figure 4-8 shows the performance of these methods as the saliency map strength is increased. As the strength increases, there are fewer confusing scenes, and the correct hand is tracked more consistently.

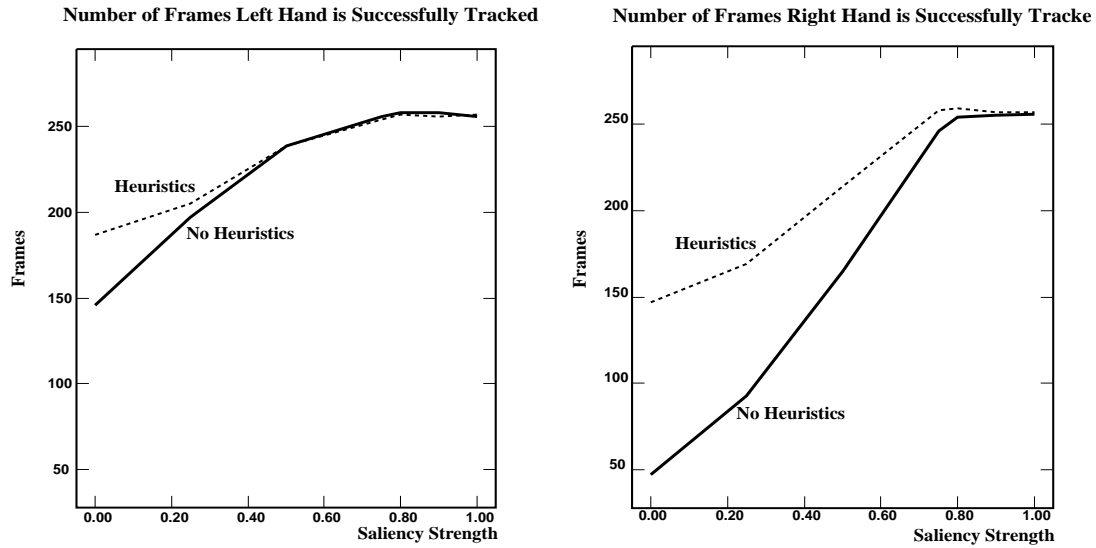


Figure 4-8: Number of times left and right hands were tracked, as function of strength of saliency map (0.0 indicates no saliency, 1.0 indicates that only the saliency map should be used.). Left: target is to track left hand. Right: target is to track the right hand.

Although the networks which use a saliency map perform significantly better than those which do not, mistakes are still made. The mistakes can be divided into three groups:

1. The hand leaves the camera's field of view. Heuristics which keep attention focused on the hand's last known location may resolve some of these cases. If the hand leaves the field of view for long periods of time, however, these heuristics will degrade.
2. If a hand is stationary for too long, when it finally moves, it leaves a bright "trail" in the MAI (see section 3), which looks like a hand. For example, see Figure 4-1.
3. The hands cross each other in the testing sequence. In general, trajectory information may help here, as long as the hands continue in the

same direction in which they started. Unfortunately, it will not help in this particular sequence because in several sequences when the hands are separated after coming together, they are moving in the opposite directions from which they started.

5. CONCLUSIONS AND FUTURE EXPERIMENTS

This chapter presented a method of incorporating domain-specific knowledge to focus attention. Domain-specific knowledge has the potential to significantly reduce training times and also to greatly improve the quality of what is learned. The presented technique is useful in vision-based tasks which involve feature tracking and detection. Often the knowledge of the constraints about the environment in which the tracking/detection is done can be used to limit the portions of the input scene which need to be processed. Although a network can learn these constraints by developing a saliency map based on future input prediction, it is desirable to incorporate these rules directly. In general, if rules or suggestions exist which provide information on the importance of the input units, a saliency map can provide a mechanism to incorporate this knowledge into the neural network.

The prediction model used for the hand-tracking task was very simple. This domain was chosen because it isolated the process of explicitly controlling the saliency map from other complicating factors which may arise in more intricate domains. However, the methods presented can be used to focus attention with more complex models in the hand-tracking task, as well as tasks in different domains. For example, in the hand-tracking task, only the single previous location was used; more sophisticated models which use hand velocity, sampling rates, and Kalman filters to combine predictions and measurements could have been used instead. In other domains, once features are found in an image, their position in conjunction with their attributes can indicate where to focus attention. As mentioned in Chapter 3, one project in which this model of attention can be utilized is detecting cars in visual scenes. Once a car is located, not only does its position reveal information of where to look next, but its orientation also provides information. To track the car, once the direction the car is facing is known, we do not need look to either side of it, but only to its front and rear. In summary, attributes, or higher level rules which

reveal position information, can be incorporated in the saliency map. This avoids the need for re-learning all of the environment's constraints. Other studies have tried to explicitly transfer knowledge between multiple neural networks; they will be briefly reviewed in Chapter 7.

The saliency map can be used as a tool for interaction with other knowledge sources. The knowledge sources can either create or augment the saliency map. One can imagine being able to interactively "point at" the saliency map to provide suggestions about where the network should devote attention. Interaction with other knowledge sources is useful in both training and operation. In training, focusing attention can help find the important features and develop the correct feature detectors. In operation, the saliency map can be a method to provide suggestions of where to pay attention, or as a method to provide instantaneous corrections when they are available (as done in this study).

For the task described in this chapter, methods other than ANNs may be used. If the filtering/prediction is accurate, simpler techniques, such as measuring the centroid of the activation in the filtered difference image, may reveal good results on the easier test sets. However, on the harder test sets, these simpler methods may not perform as well. Difficult cases for these simple operators are shown in Figure 4-7 (Bottom) in which in some examples only an outline of a hand is showing, the trail left by the hand is bright relative to the hand, or another object appears close to the hand, even in the difference image.

In future studies, it will be interesting to investigate the integration of ideas from both the hand-tracker presented in [Nowlan & Platt, 1995] and the one presented here. They used two separate convolutional ANNs for intensity and differences images with a rule-based integration of the multiple network outputs. The integration of this expectation-based system should improve the performance of at least their difference-image ANN. On the other hand, the integration of an intensity-image based ANN may improve the performance of the system presented here.

CHAPTER 5

REVERSING THE ROLE OF EXPECTATION: ANOMALY DETECTION

To this point, the experiments have required the removal of the unexpected features from the inputs. In the experiments presented in this chapter, in which anomaly detection is considered, the unexpected features are the points of interest. Therefore, attention is used to emphasize the unexpected features, while the expected features are de-emphasized. The application explored in this section is the detection and diagnosis of plasma-etch anomalies in the fabrication of semiconductor wafers.

1. REVERSING THE ROLE OF THE EXPECTED

In Chapter 2, expectation-based focus-of-attention was explored in the context of an ANN vision-based system for lane-marker tracking. In the expectation-based system, in each step, the expectation of the next-input scene was computed. These expected inputs were used to filter the actual inputs seen in the next time step. Anything which did not match the expectations was removed from the input image. This filtering removed extraneous features like old lane-markings, guardrails, or distractions in the periphery. This type of filtering can be used to pre-process the inputs to a road-following module or lane-tracking module, in which the important features appear in predictable locations.

For an obstacle-avoidance module, input filtered in the above manner is useless, since the filtering can also remove cars or other unexpected, but potentially important, features from the visual input. The second use of expectation is *to emphasize the unexpected by de-emphasizing the portions of the input which closely match that which was expected*. In this mode, the unexpected obstacles in the scene are emphasized. Input filtered in this manner can be used as input into a separate network or driving module which performs object detection/recognition and determines the appropriate action for avoiding the obstacle. Although an obstacle avoidance system could have been designed in the manner described above, there are better methods for obstacle detection, such as a radar or laser-range finder. In this chapter, an alternate task which uses the same attention methods as described for obstacle avoidance is presented. This task is the detection of faults in the plasma-etch step of wafer fabrication.

In the next section, an introduction to the problem of fault detection in the plasma-etch step of semiconductor wafer fabrication is given. In Section 3, the ANN based methods and the specifics of the experiment are given. In Section 4, five ANN based approaches to fault detection and classification are discussed. Section 5 explores in detail the most successful methods, which are based on prediction and classification, and explains why these approaches works better than the other methods used. Finally, Section 6 concludes this chapter and provides suggestions for future research.

2. INTRODUCTION TO THE PROBLEM

Plasma etch is one of many steps in the fabrication of semiconductor wafers [Russ, 1985]. Currently, fault-detection/diagnosis for this process is done primarily by visual inspection of graphically displayed process data. The plasma chamber is fitted with sensors. Wafer by wafer, time-series waveforms depicting the etch process are displayed on a computer screen. By observing these waveforms, experienced technicians can learn to detect and classify many types of faults. Because the detection and diagnosis of faults by this method is an intrinsically unreliable human endeavor [Maxion, 1996][Pope, 1986], and because of the high cost of a mistake,¹ automation of this process would make an important contribution toward increasing the accuracy, reliability, and cost-effectiveness of this procedure.

In this chapter, artificial neural networks (ANNs) are used for detecting and diagnosing fault conditions in the plasma-etch step by analyzing data gathered from etch-tool sensors. The automation of this process is complicated because of the nonstationary nature of the etcher. As the etcher produces more and more product, contaminant buildup and electrode wear alter the characteristics of the etch process, resulting in constantly changing waveforms. Over time, the waveforms can change significantly, thereby making exact template matches impossible.

In this chapter, several methods for the detection and classification of faults using ANNs are presented. Two common applications of artificial neural networks are: (1) classification of inputs into a set of classes [Fogelman-Soulie, 1995]; and (2) forecasting future values of observances [Ungar, 1995]. The first approach follows the standard classification techniques often used with ANNs. The second approach combines prediction techniques with classification techniques; this is possible because of the sequential/temporal nature of the plasma-etch process. Both approaches will be described in detail, and experiments using each approach will be presented.

1. As an example of the cost of a mistake, note that hundreds of chips, costing several hundred dollars each, may be placed on a single wafer. Therefore, an etch error, which destroys a single wafer, can cost tens of thousands of dollars.

3. EXPERIMENT SPECIFICS

In this section, the specifics of the experiment are given. Details about the plasma etcher, the types of faults to be detected, and information about the gathering and usage of the data are provided.

3.1 The Plasma Etcher

The etcher used in these experiments (a Lam 490 from Lam Research, Fremont, CA) is in active use in a production wafer-fabrication plant. This etcher is a single-wafer, plasma-etch tool system designed for use in mass production of commercial semiconductor devices. There are over 1500 of these tools installed in integrated-circuit manufacturing facilities worldwide. The LAM-490 etcher used in the present work is employed primarily for etching four-inch polysilicon wafers using a chlorine/helium chemistry. Wafers are loaded into the front of the etch tool, passing singly through the process chamber where reactive gases, ignited by an RF plasma, etch the film on the front of the wafer to define circuit elements.

Two important attributes about the plasma-etch process are of concern in this study. First, as each wafer is etched, a small amount of contaminant is deposited onto the sidewalls of the etch chamber. Second, as the plasma is repeatedly ignited, the electrodes in the chamber erode slightly. The effects of contamination/erosion become significant over the etching of many wafers. The etcher is cleaned regularly (termed the clean-cycle), and the electrodes are replaced regularly (termed the electrode cycle). The relevance of the contamination and degradation for this experiment is that the waveforms for the wafers, even those that are fault free, will change over time. Therefore, any successful fault-detection method must be able to account for these drifting effects.

3.2 Data

The data-collection phase of this experiment began on October 25, 1994, and continued until April 4, 1995. During this period, eight clean cycles were completed – an average of about 20 days/clean cycle. The total number of wafers etched during this time was 46,638 –with an average of 5830 wafers per clean cycle. The wafers were loaded in batches to be

etched, termed “runs”, which contained 1-25 wafers. Wafers were etched individually; data were gathered from multiple sensors for each etch. For the experiments conducted here, only a single sensor was used, which measured the intensity of light emitted from the plasma at the 520nm wavelength. Each etch was sampled once a second, providing approximately 140 samples per wafer.

In order to gather fault-wafer data, faulted wafers were injected periodically into the production stream. Each group of fault wafers contained one wafer of each of the four types of faults to be detected (the faults are described in detail in the next section). These injections occurred approximately every 300 wafers etched. Because of the nonstationary aspects of this process, the fault wafers injected early in the clean cycle will have different waveforms than those injected later. For this study, only one clean cycle was examined, in which there were 4052 wafers etched without faults, and 56 fault wafers injected. The problem of detecting and classifying faults is exacerbated by two factors. First, there is a very small sample of fault-wafers. This problem is partially addressed by methods which use the differences between the current wafers and previous wafers to make classifications of the current wafer. For example, if a fault-injection run was made at *time* (a), realistic training examples can be created by pairing each wafer in a with each of the wafers in the run made at *time* ($a-1$). In fact, these training examples are perhaps the most important, since faults at *time* (a) are hard to detect when they appear close to other wafers etched around the same time. For methods which are not based on comparisons, the issue of limited training data is harder to overcome. A second difficulty is that some of the wafers labeled “no-fault” may contain faults. In the preliminary analysis of the data, it was found that these unlabeled faults are mostly of different types than the four which are to be detected in this study. Complete information about the number of these other faults is not currently available. However, once the faults are labeled, the methods described in this chapter can be applied to the detection of these faults in addition to the four faults currently examined.

3.3 Faults to Be Detected

The four faults to be detected were chosen because of their occurrence in manufacturing

practice. In the wafer etch process, some of the essential steps include adding a photore-sist coating, exposing the wafer, and developing the wafer. All four fault types typically result from human error. Examples of the waveforms for each of the four fault types are given in Figure 5-1 and Figure 5-2:

1. *Unexposed or undeveloped*: a wafer which was either unexposed or undevel-
oped during the photo process.
2. *Missing resist*: a wafer which was skipped in the photo-spin step.
3. *Previously etched*: a wafer which was etched a second time.
4. *Wrong recipe*: a wafer which was etched using the wrong process recipe.

3.4 Artificial Neural Network Training & Testing Specifics

For these tests, five experimental methods and their associated network architectures are described. The learning rate and momentum parameters were held constant across all tests.

The total number of samples provided for this experiment was 4052 no-fault wafers and 56 fault wafers - 14 wafers of each type of fault. The training set is very heavily biased towards the no-fault wafers. To compensate for this, the distribution of pattern presenta-tions to the ANNs were weighted towards the fault wafers. This helps ensure that the network does not develop a bias towards the no-fault wafers due to the number of times the no-fault wafers were presented.

For training the networks, the first 36 fault wafers were used. This left the last 20 fault wafers for performing tests. As mentioned earlier, to test the detection algorithms accu-rately, the test faults must be found within the surrounding batch of no-fault wafers. It should be noted that the discussion of “surrounding wafers” only makes sense if *state of the machine* is considered. If, however, detection/classification can be done without con-sidering the state of the machine (for example, by just using the waveform of the current wafer as input, and outputting the classification), the notion of “surrounding wafers” is not applicable. Examples of both types of classifiers will be described in the next section.

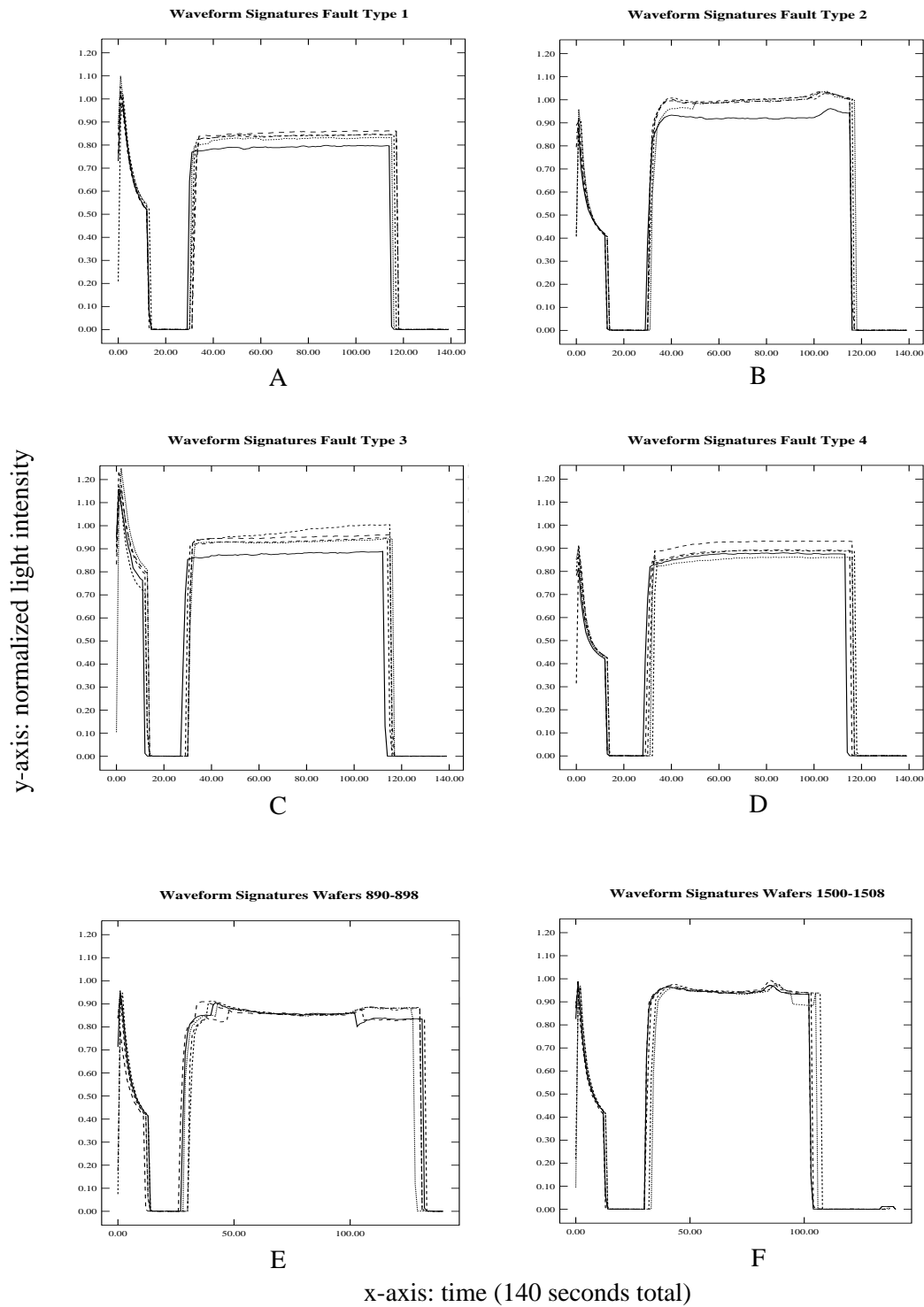
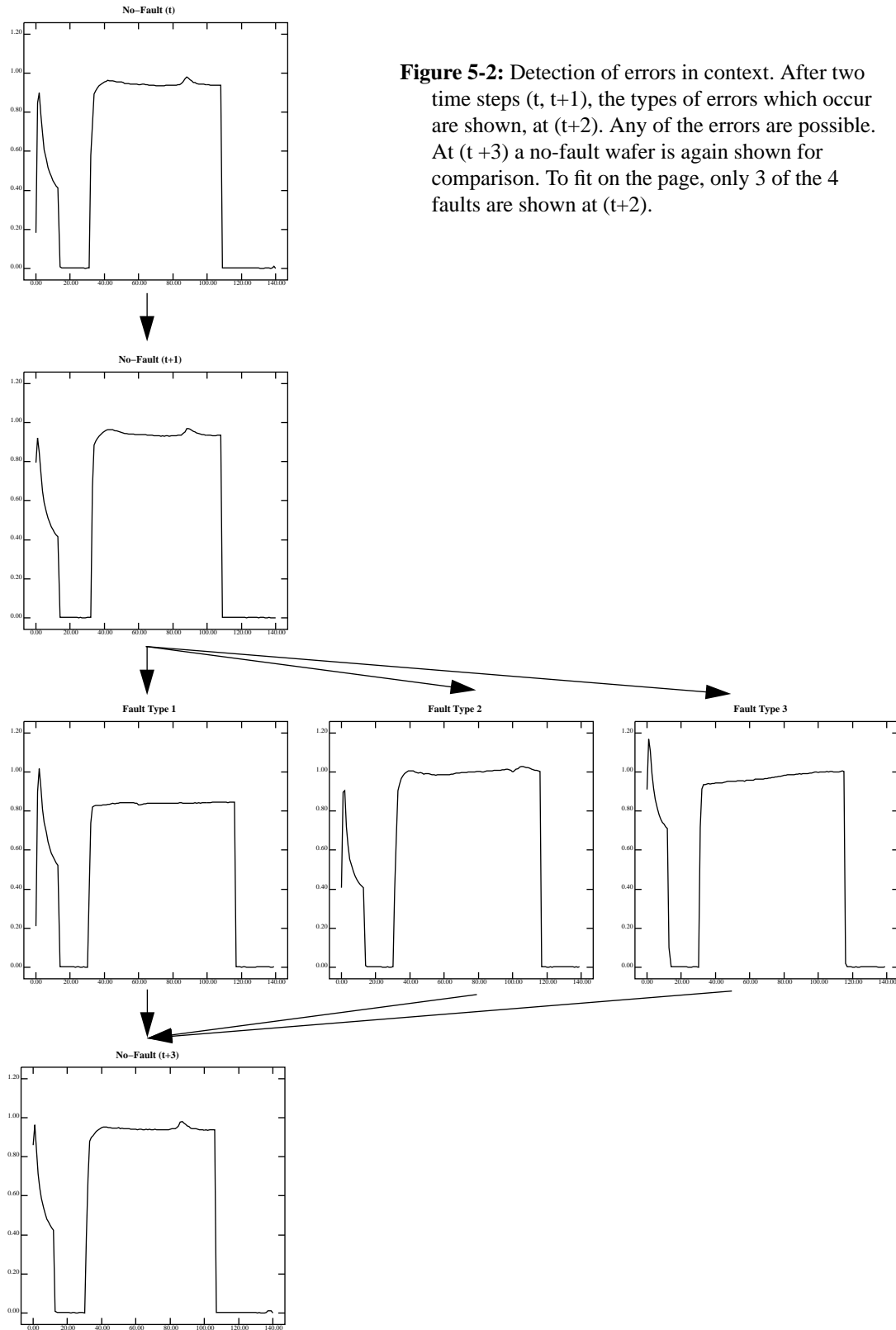


Figure 5-1: A-D: Composite wafer signature of fault-wafer types 1-4. **E-F:** Wafer Signatures of No-Fault Wafers gathered at two different times in the clean cycle. 5 wafers are shown in each graph.



In the cases in which machine state is considered, the 36 fault wafers used for training corresponded to the first 2641 no-fault wafers. The remaining 1411 no-fault wafers were used for testing. A more detailed description of the testing procedure will be provided in the next section.

Selecting the first 36 faults for training makes the detection/discrimination task harder than selecting 36 faults uniformly from the set of all 56 faults. Because of drift in the machine, the faults gathered early will have different signatures than those gathered later. Therefore, generalization to all of the faults using only first 36 faults may not be as accurate as using 36 faults spanning the entire set of faults. However, the training method presented here is more realistic in terms of actual implementation. In the future, tests to measure generalization across clean cycles will also be conducted.

4. METHODS FOR FAULT DETECTION & CLASSIFICATION

In this section, five artificial neural network (ANN) based methods for detecting the four faults enumerated in Section 2.3 are presented. Numerous network architectures were attempted for all of the methods. The best architecture found for each method is used for comparison.

In these experiments, there are two criteria which must be minimized: the number of missed faults and the number of false alarms (no-fault wafers which are erroneously identified as faults). Alone, either of these criteria can be trivially optimized. The performance of each method is reported when the sum of the percentage of missed faults and the percentage of false alarms is minimized. Although not attempted here, non-uniform weighting of the importance of missed faults and false-alarms can be used.

4.1 Method A: Base-Line

To provide a base-line for comparison to the other methods, a simple fully-connected single hidden layer network was used. This network had 140 input units (the complete waveform signature of a single wafer) and 5 outputs, representing no-fault and faults 1-4. The wafer's waveform is used as input, and the output with the highest value is considered to be the network's classification of the wafer. Although this network was able to

classify correctly all of the 20 test-set faults, it often misclassified no-fault wafers into one of the fault categories. The most frequent misclassification occurred in fault-class-2. As will be seen throughout this chapter, the no-fault wafers and the fault-class-2 wafers were often confused. The complete confusion matrix is given in Table I.

Table I: Classification Results: Simple ANN Base-Line (Method A)

		Actual Class				
		No-Fault	1	2	3	4
Predicted Class	No-Fault	663 (47.0%)	-	-	-	-
	1	10 (0.7%)	5 (100%)	-	-	-
	2	563 (39.9%)	-	5 (100%)	-	-
	3	156 (11.1%)	-	-	5 (100%)	-
	4	19 (1.3%)	-	-	-	5 (100%)

Due to the very high number of false alarms (53%), a system which automatically monitored the etcher based on this simple method would not be usable. The number of false alarms can be improved if we are willing to allow more missed faults. For example, if we allow a single missed fault (by stopping the training at a different point), the false-alarm rate decreases to 47%. As the number of allowed missed faults increases, the number of false-alarm errors decreases.

For comparison, simpler methods were also attempted. Instead of using an ANN with a hidden layer, a simple perceptron was also tried. This resulted in a 20% miss rate and approximately a 40% false-alarm rate. A simple fault detection method (not classification) is to measure the difference between the average “no-fault” vector and the input vector to be classified. By varying the threshold, the system can be made more or less conservative, with respect to the number of faults signalled. Two hundred uniformly spread threshold settings, between 0 and the maximum difference between the average no-fault vector and the fault vectors, were used. The smallest error (missed rate + false-alarm rate) was a 14% miss rate and 41% false-alarm rate.

4.2 Method B: Using the Previous Wafer for Comparison - Incorporating Machine-State Information

As mentioned earlier, one of the known problems with plasma-etch tools is process degradation from chamber contamination and electrode erosion. Because these data were collected over many days, in which many etches were made, the changing conditions of the etcher must be considered. One method for incorporating state information is presented in this section.

The current state of the etcher can be considered by using the signature of the last known wafer which did not contain any of the four faults. This signature can be used as a basis with which to compare the current wafer's signature. Using the differences between the current wafer's signature and the previous wafer's signature provides a simple way to incorporate state into the decision process. An implementation of this idea is as follows: the classification neural network is given both signatures as input. Further, in order to accentuate the differences between the two input wafers, the point-by-point differences are also input. Although using the differences is strictly not necessary, since the ANN should be able to compute these differences if required, it was found to improve learning speed and accuracy. The drawback of this method is that it triples the number of inputs.

The first network architecture used was a standard, fully-connected network, similar to the one described in Method A, with 3×140 input units. However, this network's performance was poorer than that of many others; the best architecture found is described here. This network architecture used a separate hidden unit for each set of 3×5 input units, as shown in Figure 5-3. The motivation for this architecture was to allow each hidden unit to specialize in analyzing only small portions of the input. The hope was that there would be recognizable features in these small portions. Similar architectures were used in Chapter 3, and will be used again in Chapter 6.

As described previously, in Section 2.2, a fault could have been injected into any of the 25 positions in the run. Since training this network requires pairs of wafers as input, examples were created by pairing all members in the run made previous to the fault run with those in the fault run. Therefore, unlike Method A, in which only 20 fault wafers were tested (since no state information was used), many more pairs can be tested using

Method B. The total test set size is 396 fault-wafer pairs¹, as well as the 1411 no-fault wafer pairs. This method achieved a miss rate of 18.4% and a false-alarm rate of 5.6%. A more detailed breakdown of the results is given in Table II.

Table II: Classification Results: Single Network with State Information (Method B)

		Actual Class				
		No-Fault	1	2	3	4
Predicted Class	No-Fault	1332 (94.4%)	36 (36.4%)	33 (33.3%)	-	4 (4.0%)
	1	1 (0.1%)	63 (63.6%)	-	-	-
	2	72 (5.1%)	-	66 (66.7%)	-	-
	3	4 (0.3%)	-	-	99 (100%)	-
	4	2 (0.1%)	-	-	-	95 (96.0%)

Tests similar to the ones described in the previous section were also conducted here. Instead of using an ANN with a hidden layer, a perceptron was tried. This resulted in degraded performance. Also, instead of using an ANN-based method, detection of faults based on the magnitude of the summed point-by-point differences between the previous

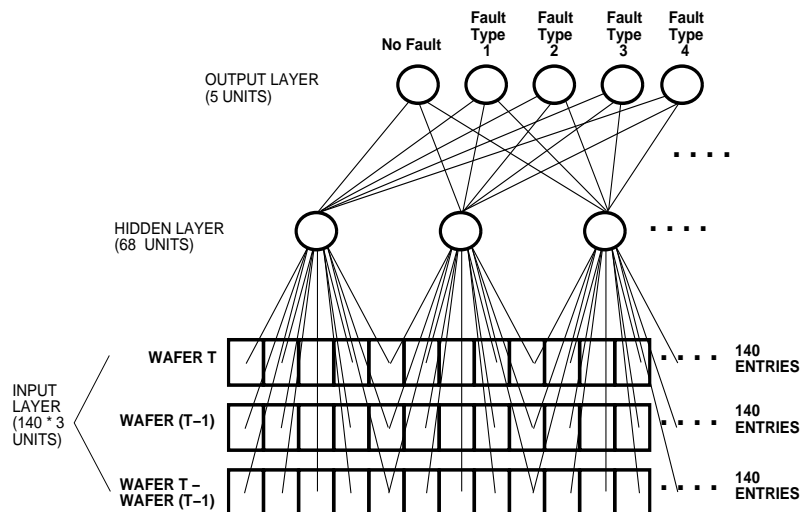


Figure 5-3: Network architecture used for Method B.

1. This number is not greater because not all of the runs contained 25 wafers.

no-fault wafer and the wafer to be classified was attempted. Again, 200 threshold values were used, uniformly spanning the smallest distance between input pairs and the largest difference between input pairs. For the same percentage of missed faults (18.4%), the false-alarm rate was approximately 10%.

4.3 Method C: Using State & Individual Networks

In this set of experiments, rather than using a single network to predict the fault type of a particular wafer, as described in the previous section, five networks were used: four to detect each of the four fault types, and one to detect wafers which did not contain these faults. Each network was trained to produce an output of +1.0 only if the input wafer was of its assigned fault type. If the wafer was of a different fault-type (including no-fault), the network was trained to output a -1.0. The inputs were the same as used in Method B. There was only 1 output per network. Each network's output was interpreted as classifying the input wafer as belonging to its class if the output was a value greater than 0.0. If the value was less than 0.0, the output was interpreted as a classification outside the network's trained class. If more than one network classified the wafer to be in its class, a method for arbitration would be required. However, in the tests performed, this situation never arose.

The four networks trained to detect fault types 1-4 performed well. However, the network trained to recognize no-fault wafers often did not recognize no-fault wafers (20% error), and often responded positively to wafers which had faults (39% error). Nonetheless, because only 5 classes need to be discriminated, using 5 networks is redundant. Because of the accuracy of the networks 1-4, the system can still work without the no-fault recognition network. If networks 1-4 all respond negatively to the input wafer, then the wafer is classified as a no-fault wafer. This method reduces the percentage of missed faults to 12.1%, and reduces the percentage of false alarms to 1.5%. More details on the performance of each network are given in Table III. Note that a confusion matrix cannot be given in this case as each network only indicates whether it believes each example is or is not a member of the class it was trained to recognize. The key to the table is given below:

- **Correctly Classified Faults:** These wafers have faults of the type the network was trained to recognize, and was successfully able to recognize. (Example: Network 1 successfully recognizes Fault 1.)
- **Misses:** These wafers have faults of the type the network was trained to recognize and was *not* able to recognize. In practice, this type of error may be the most damaging. (Example: Network 1 does not recognize Fault 1.)
- **Correct Other Faults:** These are wafers manifesting faults other than the ones the network was trained to recognize. This is the number of “other-faults” to which the network correctly responded negatively. (Example: Network 1 successfully rejects Fault 2.)
- **Misclassified Faults:** These are the same types of wafers as above, fault wafers of a different kind than the network was trained to detect. This is the number of wafers the network mistakenly took to belong to the class on which it was trained. (Example: Network 1 erroneously accepts Fault 2 as Fault 1.)
- **Correct No-Faults:** This is the number of no-fault wafers to which the network responded correctly. (Example: Network 1 successfully rejects no-fault.)
- **False Alarms:** This is the number of no-fault wafers that were classified as faulty. (Example: Network 1 erroneously accepts no-fault as Fault 1.)

Table III: Classification Results: Multiple Networks with State Information (Method C)

		Correctly Classified Faults	Misses	Correct Other Faults	Misclassified Faults	Correct No-Faults	False Alarms
Network	1	60 (60.6%)	39 (39.4%)	297 (100%)	0	1411 (100%)	0
	2	98 (99.0%)	1 (1.0%)	297 (100%)	0	1390 (98.5%)	21 (1.5%)
	3	99 (100%)	0	297 (100%)	0	1411 (100%)	0
	4	91 (91.9%)	8 (8.1%)	297 (100%)	0	1411 (100%)	0

4.4 Method D: Computing Expectations

When humans monitor the etch tool visually, or when they perform many other types of visual scene analysis, one method of eliminating extraneous, or irrelevant, information is through filtering their input based upon expectation. In this thesis, expectation was used

in an artificial neural network-based system to filter noise and irrelevant features from an input visual scene. This was accomplished by suppressing the portions of the scene which did not match a computed expectation of the next scene. In the domain explored here, however, the portions of the input which are anomalous are important. Since the task being performed requires anomaly detection, expectation can be used to *accentuate the differences* between the expected inputs and the actual inputs. More details are given below; these ideas are returned to in Section 4. Using expectations for fault detection has also been explored in [Maxion, 1990].

Expectation is computed as follows. Given the waveform signature of wafer(T-1), an expectation of wafer(T) can be formed. This expectation is formed using a second neural network. The input to this second neural network is the waveform signature of wafer(T-1); the output is the expectation of the signature of wafer(T). This network is trained in a supervised manner by using the wafer's waveforms in sequential order. The target output for each example is the next wafer in sequence. The target output is the length of the input (140 units). This process is related to auto-encoding networks; however, in the present method, the *next* time-step is predicted. Somewhat similar prediction methods have been explored in [Tebelskis, 1995] (Tebelskis' work is described in more detail in Chapter 7). The network is trained only to predict the waveform of the next no-fault wafer; the network is not trained to predict fault wafers.

Like the training of the lane-tracking system in Chapter 2, we can modify the training procedure to discourage the memorization of specific transitions. The training is augmented as follows: rather than only predicting the waveform for the next wafer, some training examples with input wafer(T-1) are given the waveform of wafer(T+1) (rather than wafer(T)) as the desired output. This helps to ensure that the network does not memorize specific transitions. The prediction network architecture is shown in Figure 5-4.

Based upon the expectation of the waveform of wafer(T) (termed *expectation(T)*), wafer(T), and the differences between *expectation(T)* and wafer(T), a classification of the wafer is made. The classification network is the same architecture network as the one used in Method B. However, in Method B, wafer(T-1) was used as input, while in this

method expectation(T) is used as input. This method slightly reduces the percentage of missed faults to 9.3%, but there is an increase in the percentage of false alarms to 10.1%. The results for this method are shown in Table IV.

Table IV: Classification Results: Computed Expectations - Single Network (Method D)

		Actual Class				
		No-Fault	1	2	3	4
Predicted Class	No-Fault	1268 (89.9%)	9 (0.9%)	28 (28.3%)	-	-
	1	5 (0.3%)	90 (91.0%)	-	-	-
	2	125 (8.9%)	-	71 (71.7%)	-	-
	3	4 (0.3%)	-	-	99 (100%)	-
	4	9 (0.6%)	-	-	-	99 (100%)

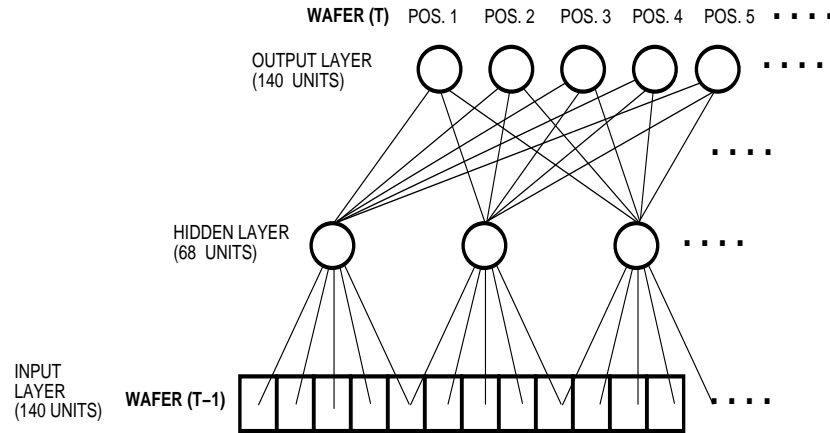


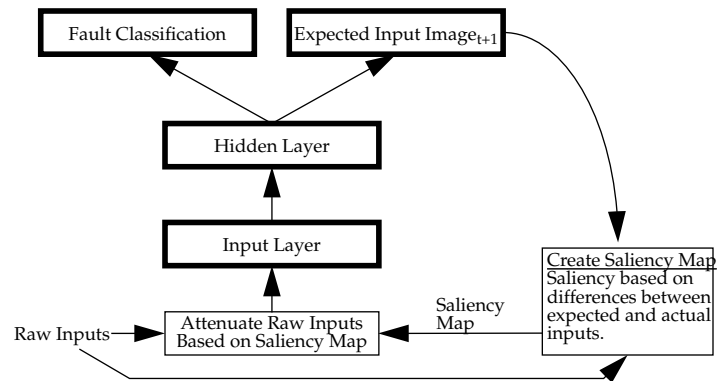
Figure 5-4: Network architecture for creating expectations of the next input waveform.

4.4.1. A New Use of Expectation

The use of expectations described in this section is quite different than explored previously in this thesis. In comparison to the model in Chapter 2, there are two key differences. First, the expectations were not used to create filters for the subsequent inputs.

Second, the expectations were created from a separate neural network; therefore, they are not task-specific in the same manner as described in Chapter 2. These differences are addressed below. See Figure 5-5 for a schematic of the architectures used.

The network architecture in Ch.2-shown for anomaly detection. Predictions about the next input image are made based upon the contents of the hidden layer.



The network architecture used here.

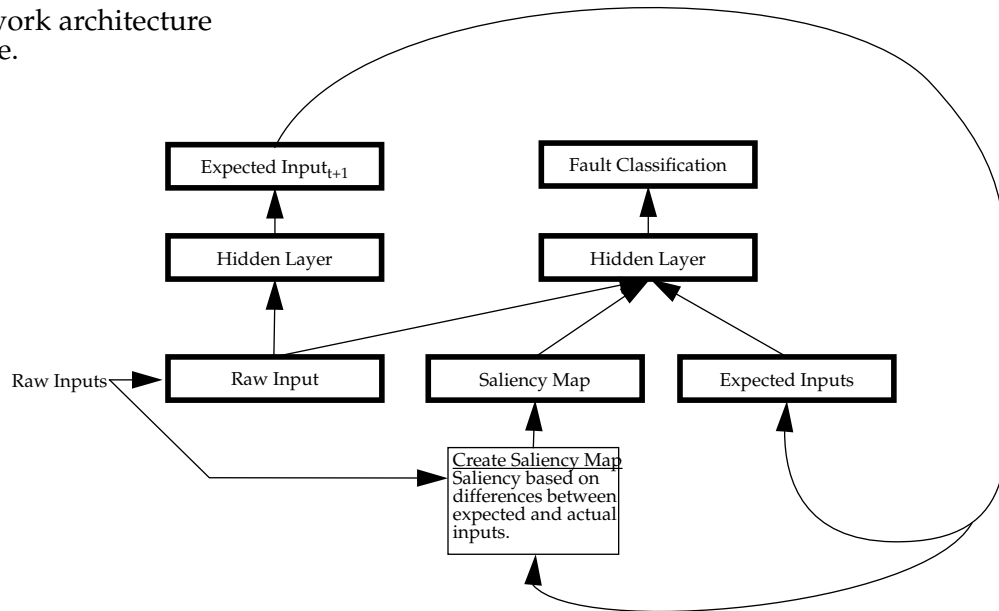


Figure 5-5: Top: architecture adapted from Chapter 2 for anomaly detection - see text for why this may not work well. Bottom: The architecture used here.

Like the previous applications, expectations were used to create a saliency map. For

anomaly detection tasks, the saliency map is the difference vector between the expected and actual inputs. In some anomaly detection tasks, it may be sufficient to use this saliency map with the filtering procedures described previously to de-emphasize the *expected* portion of the input. However, in this task, these methods are not appropriate. De-emphasizing or removing portions of the input would yield poor results since anomaly detection in this task relies upon the relative magnitudes of many inputs. In the preliminary experiments tried, de-emphasizing only a few of the inputs created misleading artifacts in the input. Therefore, instead of modifying the original inputs, the expectations were used as extra inputs to the network. Additionally, the difference vector between the expected and actual inputs, which is the saliency map, was also used as input. The detection network can come up with arbitrarily complex strategies to use these extra inputs in the regions required, while ignoring them in other regions.

The network architecture used for this experiment was quite different than previous experiments. Since the main task is to detect anomalies, the network should only encode information about the regions of the input that often contain anomalies. Although the task forces the network to concentrate on *anomalies*, information about the *regularities* is required for accurately creating expectations of no-fault wafers. Note that we are only interested in being able to predict no-fault wafers accurately since it is impossible to predict some of the errors that we examined (for example, predicting, based upon the current wafer, that the next wafer will have been previously etched). Therefore, expectations are created to predict the next no-fault wafer, since this will be used for comparison with the actual input waveform in the next time step. The hidden units of the detection network contain the wrong information for this prediction. When a prediction is made from the information encoded, *it will be unable to accurately predict the portions of the waveform which do not often contain information for detecting anomalies*. Therefore, when this prediction is compared to the next input waveform, the differences may not indicate true anomalies. This will render the difference-based saliency map useless. To avoid this problem, a separate network is used for prediction. Although the task-specificity is not preserved, the predictions are more accurate. As will be described in Section 5, the network is only trained with no-fault examples as inputs and outputs; it is not trained to predict faults, and it is not trained to make predictions from faulty wafers.

4.5 Method E: Computing Expectations and Individual Networks

In Method E, five networks were used in a technique similar to Method C. Each network was assigned a particular fault type. Each network was trained to produce an output of +1.0 only if the input wafer was of its assigned fault type. If the wafer was of a different fault-type (including no-fault) the network was trained to output a -1.0. The inputs were the same as Method D (wafer (T), expectation (T), difference between expectation (T) and wafer (T)), and there was only 1 output. The strengths and weaknesses of this method were the same as for Method C. All of the fault detection networks worked well. However, the network that was trained to detect no-fault wafers often did not recognize no-fault wafers (26% error), and often responded positively to wafers which had faults (33% error). As was done in Method C, since four of the five networks worked well, accurate class discriminations could be made. The individual network architecture is shown in Figure 5-6. This system had the best performance overall; the percentage of missed faults was lowered to 1.3%, while the percentage of false alarms remained low at 2.3%. The results of the system without using the no-fault detection network are shown in Table V.

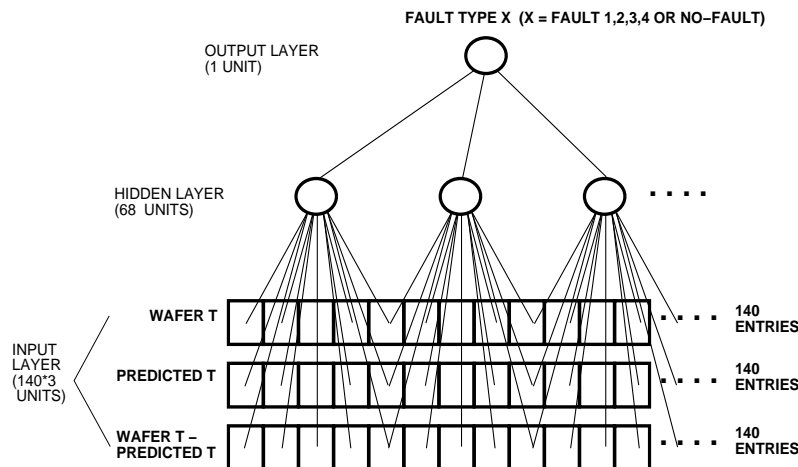


Figure 5-6: Network for Method E: 5 of these networks were trained; one for each of the four fault types, and one to indicate none of the four faults.

In Figure 5-7, the network outputs for each class of wafers are shown. These diagrams

Table V: Classification Results: Computed Expectations - Multiple Networks (Method E)

		Correctly Classified Faults	Missed Detects	Correct Other Faults	Misclassified Faults	Correct No-Faults	False Alarms
Network	1	99 (100%)	0	297 (100%)	0	1411 (100%)	0
	2	96 (97%)	3 (3%)	297 (100%)	0	1384 (98.1%)	27 (1.9%)
	3	99 (100%)	0	297 (100%)	0	1411 (100%)	0
	4	97 (98%)	2 (2%)	297 (100%)	0	1406 (99.6%)	5 (0.4%)

show how each network responds to each wafer. For example, in the graph “Network Simulation of Fault Type 1”, networks 2,3 and 4 all responded with low values (less than 0.0). However, network-1 responded with values all greater than 0.0. In the graph “Network Simulation of Fault Type 2”, network-2 responded with the highest values. In three of the 99 testing examples shown, network-2 responded with values less than 0.0, indicating missed detects. Also, on the last example, network-4 responded with a value barely less than 0.0. In the graph “Network Simulation of No-Fault”, the responses of the 4 networks to the 1411 no-fault testing wafers are shown. Ideally, each network should respond with a negative value, since these wafers do not belong in any of the classes the networks are trained to detect. Network-2 misclassified no-fault wafers as belonging to its class 27 times; network-4 made similar mistakes 5 times.

4.6 Summary of Empirical Results

The performance of five ANN-based methods for detection and classification of four faults was evaluated. A schematic diagram of these approaches is shown in Figure 5-7. The last four of these incorporate state information. The first method, the base-line, has a different scale on which to measure results because it does not take state information into account. Because of its very high false-alarm rate, it cannot be used in practice. In Table VI, the last four methods are compared in terms of misses and false alarms.

In this study, a single large network did not provide results as good as those provided by separate networks that recognize each fault type. The expectation-based methods pro-

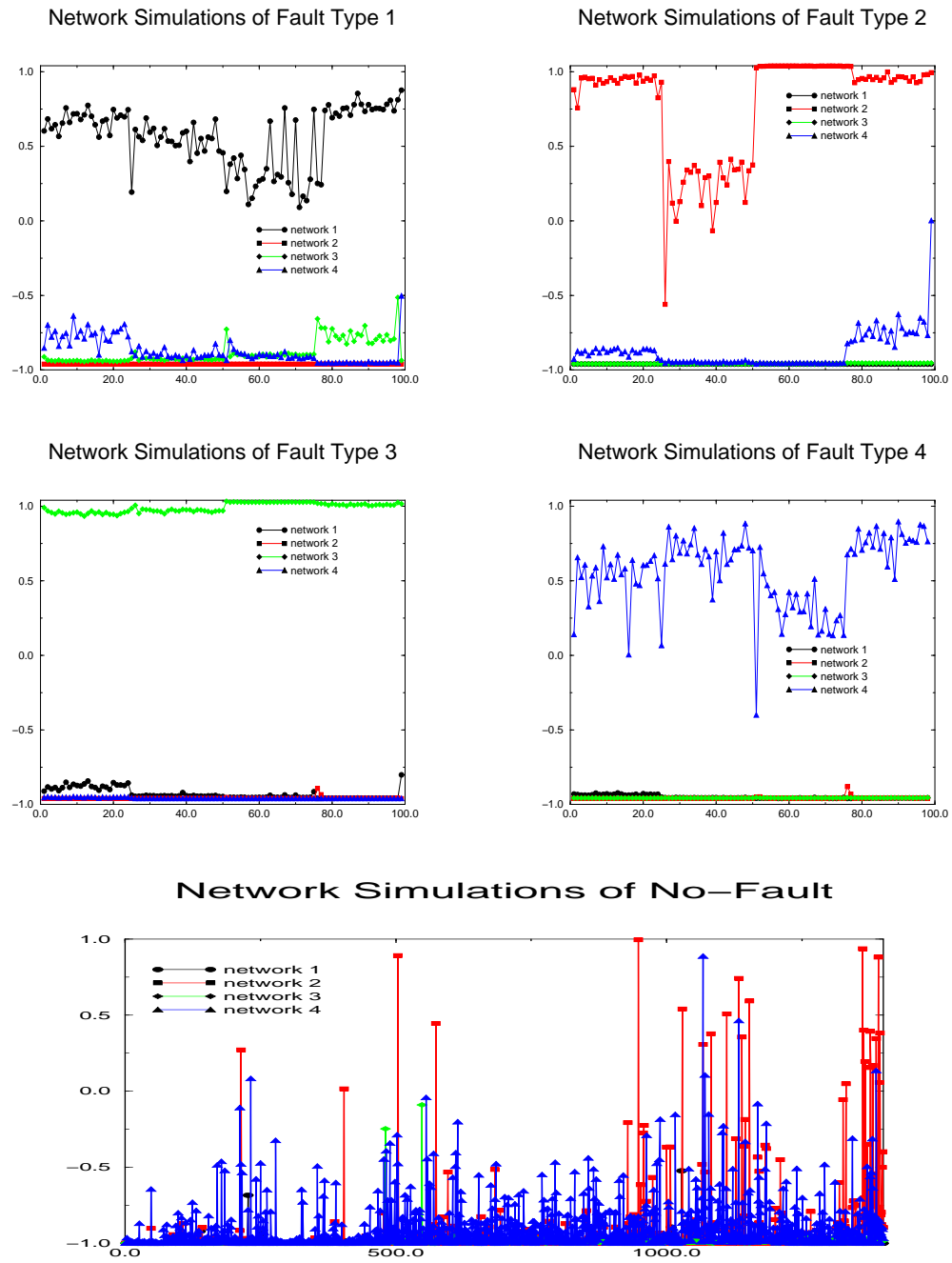


Figure 5-7: The individual network's outputs (Method E) to each fault type, and no-fault wafers. The X-Axis is the wafer number, and the Y-Axis is each network's response.

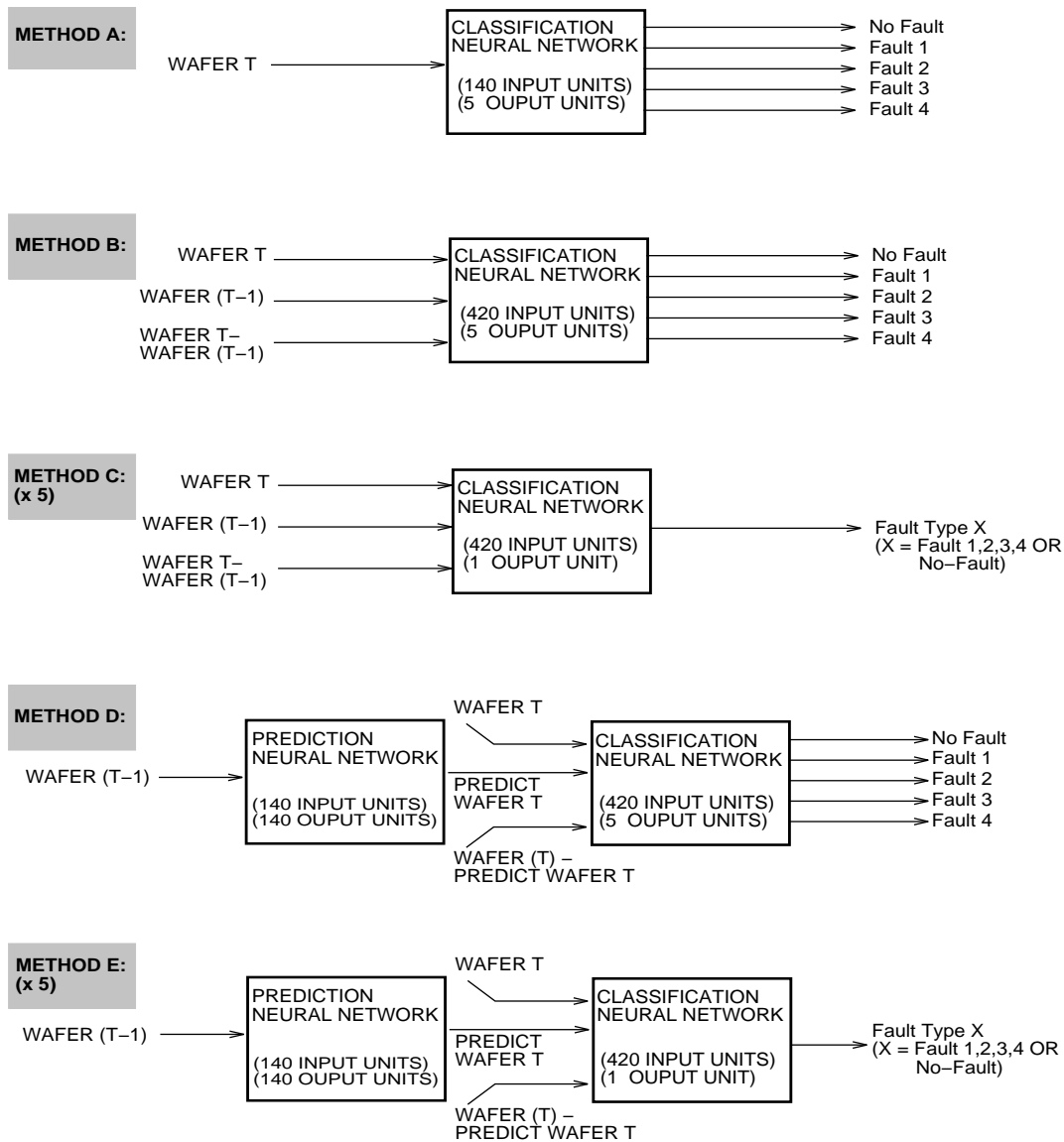


Figure 5-8: An overview of the five approaches to fault detection and classification.

vide better results, in terms of the total number of missed faults, than their counterparts which use state information in the form of the previous no-fault wafer. There is a small degradation in the number of false alarms; nonetheless, the trade-off still heavily favors the individual-network expectation-based methods because of the fewer missed faults. Although the number of missed faults can also be reduced to this level for the other

Table VI: Performance Summary

		# Classification Networks Used	Total Missed Faults (%)	False Alarms (%)
Method	B: State Info.	1	73/396 (18.4%)	79/1411 (5.6%)
	C: State Info.	4	48/396 (12.1%)	21/1411 (1.5%)
	D: Expectation	1	37/396 (9.3%)	143/1411 (10.1%)
	E: Expectation	4	5/396 (1.3%)	32/1411 (2.3%)

methods, the number of false-alarms increases beyond the level in Method E. It should also be noted that the differences between the numbers of false alarms in Method C and Method E could be due to the noise in the testing set (i.e., errors in labeling the no-fault wafers), as described in Section 2.

5. WHY USING EXPECTATION IMPROVES PERFORMANCE

For the task of wafer anomaly detection, the important portions of the input are those that *do not match expectation*, since the expectation was designed to predict only no-fault wafers. One method of emphasizing these portions is by using the differences between the expected and actual waveforms as inputs into the network. This difference vector emphasizes more heavily the portions in which the difference between the expected and actual input is large.

The use of expectation provides a means to incorporate machine state information into the prediction. However, Methods B and C also provided state by explicitly using the previous no-fault wafer's waveform. A reason that expectation is able to outperform using the previous wafer's signature is that expectation-based methods are less sensitive to the previous wafer's waveform than methods which explicitly use the previous wafer's waveform. Although the prediction of the next wafer's waveform is based upon the input wafer's signature, using the prediction often mitigates the effects of small variations in the original input wafer, thus providing a cleaner training signal. Methods which explicitly use the previous wafer's waveform for classification are subject to these variations in their inputs. See Figure 5-9.

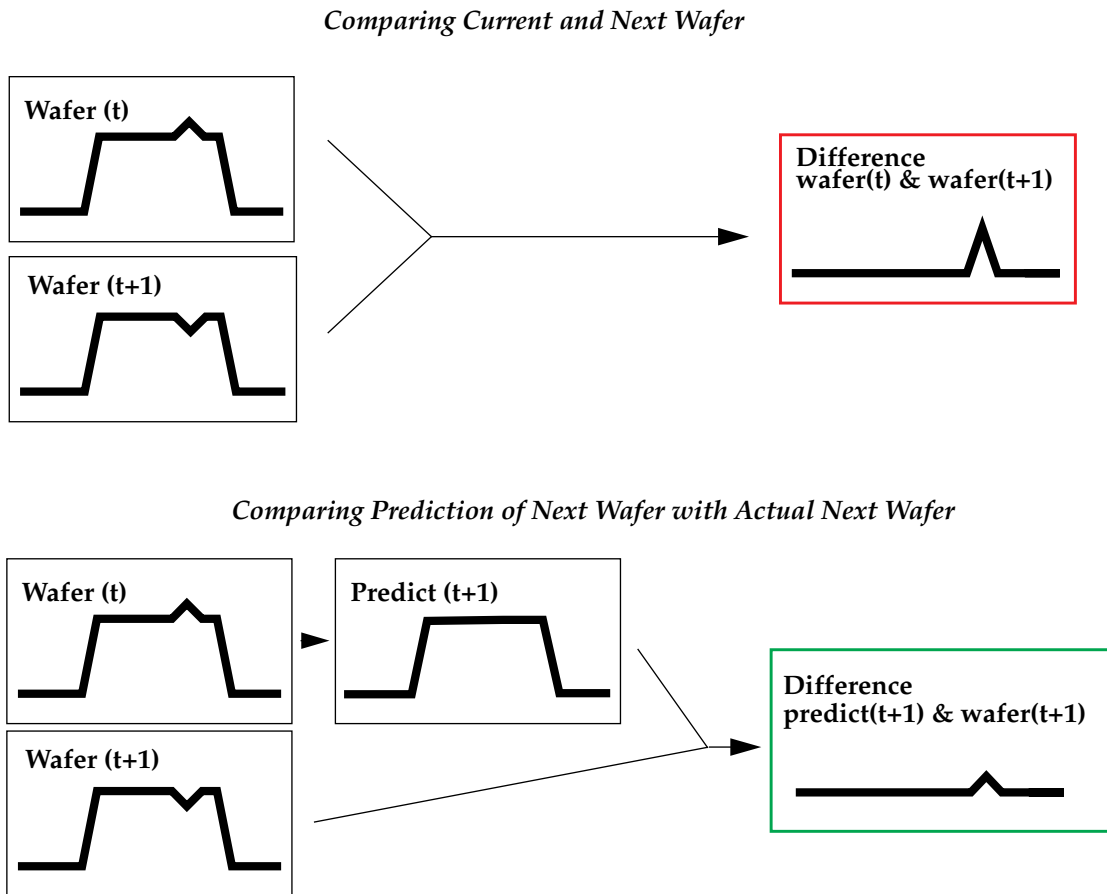


Figure 5-9: Benefit of using a prediction to compare with the input wafer instead of the previous wafer. Small differences which may not be true anomalies can be compounded when using two noisy waveforms. The resulting system is less sensitive to small anomalies in the previous wafer.

As an example of the phenomenon described above, see Figure 5-10. In Figure 5-10A, note that wafer #121 has an uncharacteristic hump starting around input 70. When wafer #120 is used as input to the prediction network, it predicts a wafer without the hump. Further, when wafer #121 is used as input to the prediction network, the predicted wafer is *not* similar to wafer #121; instead, the effects of the hump in #121 are mitigated. The expectation of wafer #122, which was derived from wafer #121, was made fairly accurately. The same types of effects are shown in Figure 5-10B: wafer #3654 has a small “notch” missing in its left top (inputs 40-55) while the predicted wafer does not; and the next prediction (of wafer #3655) is only slightly affected by this notch. In summary, using expectation de-emphasizes the small variations in the waveforms; therefore, the classification network is given cleaner training data. *It should be noted that in the training phase,*

expectations are never formed from wafers which contain faults; expectations are only formed from wafers which do not contain any of the four faults. In testing, expectations are never formed from wafers which are classified as faulty. Therefore, only anomalies in the wafers which have been classified as “not containing any of the four faults to be detected” are mitigated through the prediction process.

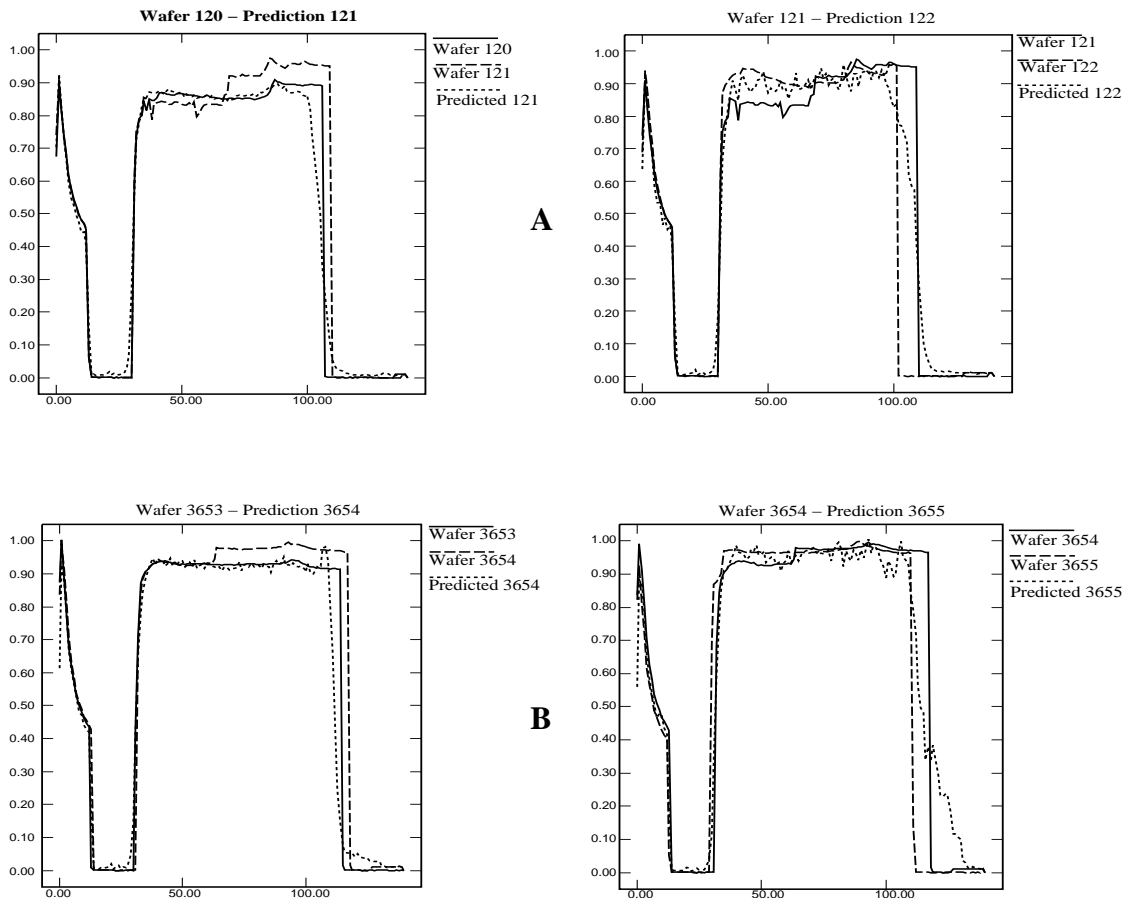


Figure 5-10: Sample Input Wafers & Next Wafer Predictions. Note the hump in wafer #121, which is mitigated in the prediction of wafer #122. The same effect is seen for wafers #3654 and #3655.

6. CONCLUSIONS & FUTURE DIRECTIONS

Due to the difficulty of controlling the plasma-etch step, and because the etch step is typically repeated several times for each wafer, attention has been devoted to automatically

monitoring the plasma-etch chamber for detection and diagnosis of faults. Currently, monitoring is done by humans who visually inspect the etch waveforms as they appear on computer screens. Since plasma etching is a commonly used process, and because of the unreliability of human detection and diagnosis, automation is important.

This chapter has presented five artificial neural network based methods for detecting and classifying faults. Because of the nonstationary conditions of the plasma etcher, which cause waveforms to change over time, the state of the etch machine must be considered when making detections and/or classifications. In this study, two methods of providing the ANN with the machine state were explored. The first method used the last previous no-fault wafer, in addition to the current wafer, as input into the ANN. The second used an expectation of the current input. It was found that the expectation-based model improved the detection and classification score. In this study, a second ANN was used for computing the expectation of the next wafer's signature. However, other neural architectures, such as recurrent networks, can be used to maintain internal state in conjunction with output classifications [Jordan, 1989][Elman, 1990].

There are four immediate directions for future work with ANN-based fault detection. First, a measure of the network's reliability is needed. As described in Chapter 2, [Pomerleau, 1994] has proposed a method to measure reliability which is based upon determining how closely the network's internal units, which have been trained to perform the desired task, represent the inputs. If the network has represented the inputs well, the reliability is assumed to be high. Another method to measure reliability is to use different training metrics which are intended to reveal confidence, such as cross-entropy [Hinton, 1987]. Both of these methods can be used to determine how certain the networks are of their decisions.

A second direction for future work is in building redundancy into the fault-detection system. For example, in the systems which use four individual networks for detecting faults, if an additional network can be trained to detect no-fault wafers it could provide an additional test to ensure that a fault does not escape the system. Although no-fault detection networks could not successfully be trained in these experiments, more data, other error minimization metrics, such as Classification Figure of Merit [Hampshire & Waibel, 1989],

or other training algorithms [Kitano, 1990][Baluja, 1996], may yield improved results.

A third direction for future work is using a moving average of the previous several no-fault wafer waveforms for comparison with each new waveform. This may yield insights and extensions to the expectation-based methods employed here.

Finally, perhaps the most important direction for future work is extending this work to the detection of novel fault conditions. Currently, only four labeled fault conditions are detected. However, in practice, the system should be able to detect anomalies on which the system has not been explicitly trained. This may necessitate the decoupling of the fault detection and diagnosis procedures. Decoupling the two procedures provides the benefit of detecting faults that were neither explicitly labeled nor available in the training set. ANNs, as well as other machine learning methods, have been applied to similar problems of novelty detection [Japkowicz *et al.*, 1995][Maxion & Olszewski, 1993]. One such system currently under investigation is the Harbinger system [Maxion, 1996]; this system uses rules derived from decision-tree methods. In the preliminary experiments applying Harbinger to this domain, the results appear promising. A way to use the ANN-based methods presented here in conjunction with the Harbinger system is to use Harbinger for the detection of faults and ANNs for the classification of detected faults.

CHAPTER 6

EXPECTATIONS WITH NON-TEMPORAL DATA

To this point, all of the selective attention mechanisms were developed to address temporal data, since expectation is most clearly defined over multiple sequential time steps. Although the main focus of the thesis will remain on filtering in temporal domains, expectations, as well as saliency maps, can be used to understand the implicit focus of attention in trained networks. Methods for extracting and enhancing the implicit saliency map are presented. The results are demonstrated in the task of face detection in arbitrary visual scenes.

1. INTRODUCTION

In visual scene analysis, the reliance on only a single feature, when many are available, can often degrade performance when the system is tested in real-world situations. However, the most common neural network training algorithm, standard error-backpropagation, often focuses on only a few of the most reliable features. This chapter presents a practical way to encourage standard backpropagation to distribute its input reliance in order to ensure that no-single feature is given too much importance.

This chapter differs from the previous ones in two respects. First, the data analyzed is not related temporally, so no explicit saliency map based upon expectations can be made. Second, the goal of the described method is often to *distribute* the focus of attention. In visual scene analysis, there may be many features which provide information for solving a particular task. In real-world tasks, a distributed input reliance is important for two reasons. First, some features may disappear from the input visual image, yet enough secondary information may be present to solve the task. Second, the false detection of objects may occur because a few features may trigger a false detection because the context of the rest of the input was not considered. For example, in the case of face detection, dark spots which appear similar to eyes may be found and a detection triggered, although a nose or mouth was not found. Examples of this will be shown throughout this chapter.

As shown in previous chapters, a saliency map can be used to emphasize different regions of the input. However, the saliency map (as used in this thesis) makes the implicit assumption that the data is related temporally, since the saliency map is computed from the inputs of time($t-1$) to focus the attention of the inputs at time(t). However, in many visual classification tasks, the inputs may not be temporally related. For example, consider detecting zip codes on envelopes or detecting whether an object is present in an arbitrary scene. In these examples, expectation based methods may not be appropriate.

Although a saliency map is not explicitly computed, a trained network's weights specify the portions of the inputs to which the network will pay the most attention. Therefore, the network uses an *implicit saliency map*. In this chapter, we present a method to visualize and modify the saliency map.

In this chapter, we examine several methods of encouraging distributed reliance of features in the input retina. Noise is added into various portions of the network during training. This ensures that all of the input features will be obscured at some time during training, thereby forcing the network to distribute its reliance across all available features. Many variations of this basic technique have previously been explored in the neural network literature. We show the effects of using several of these variants with respect to the network's focus of attention.

In Section 2, the face detection task is described, and spatially constrained ANNs are reviewed. Section 3 reviews some of the methods used to introduce noise into ANNs, and describes the methods used here. Section 4 details the experiments conducted in the face-detection domain and displays the effects of noise on the implicit saliency maps.

2. FACE DETECTION

In this chapter, the task of upright-frontal-view face detection in arbitrary scenes is explored [Rowley, Baluja & Kanade, 1995, 1996][Sung & Poggio, 1994][Valliant *et. al.*, 1994][Yang & Huang, 1994]. This is a very difficult problem; it has been hypothesized that this problem is harder than isolated object recognition [Poggio & Beymer, 1996][Hurlbert & Poggio, 1986]. The system operates as follows: a window of size 20x20 pixels is passed over a full resolution image (which can be any size), and a network is trained to tell whether there is a face in each 20x20 image. To detect faces at larger sizes, the image is reduced (in increments of 80%), and the 20x20 window passed over the reduced image. For these prototype tests, the training set contains 4160 examples which contain faces, and 5040 images which a network trained to perform this detection task (in another study) detected as false positives (indicated incorrectly that there was a face). Therefore, these images may be difficult for networks to distinguish as non-faces, as shown in Figure 6-1. The testing set contains 5600 face and 5800 non-face images. The full system does not use a static training set as is employed here. Instead, it uses an active learning paradigm for automatically selecting non-face examples. Nonetheless, the techniques explored in this section have been used for analyzing the performance of the system. Further, the effects of training with noise, which are explored in this thesis with a static train-

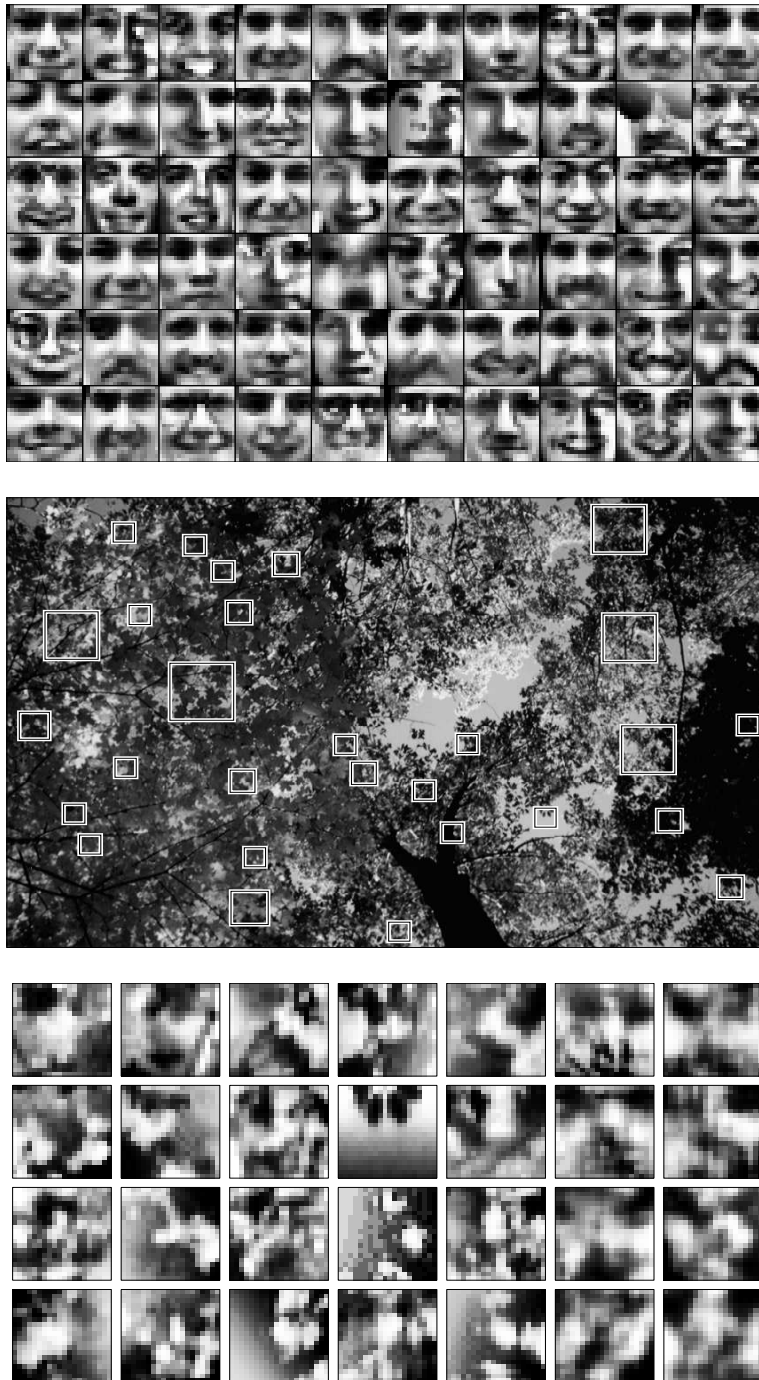


Figure 6-1: **Top:** Positive Examples. **Middle:** image with no faces. **Bottom:** Typical false positives added into the training set from mistakes made in image.



Figure 6-2: Face Detection Examples. The numbers correspond to the actual number of faces in the image/ the number of detected faces/ the number of false detections. Faces are missed in image J, and false detections are present in image A. M was provided by Sung and Poggio at MIT. These images are taken from [Rowley, Baluja & Kanade, 1996].

A note about the image layout: These images were automatically packed using the PBIL algorithm [Baluja, 1994], trying to minimize the vertical space used, given the available horizontal space.

ing set, will be also be explored in detail in the full system. The system is described in [Rowley, Baluja & Kanade, 1995 & 1996]. An example of the goal as well as some of the results of the full system are shown in several examples in Figure 6-2.

Face detection, as well as many other vision based tasks, has the property that the relevant features in the input retina are composed of pixels which are spatially localized. For example, the eye is composed of pixels which are next to each other. These types of tasks also have another important property - the important features usually occur in similar locations in the inputs in most of the frames. For example, when detecting upright faces, the eyes appear in the top half of the image. One way of incorporating these two properties into ANNs is to use a spatially constrained network architecture. In spatially constrained architectures, each of the units in the hidden layers (which have connections to the input layer) are connected only to a small number of spatially-close input units. This is in contrast to the traditional fully-connected networks, in which the hidden units are connected to all of the input units. Spatially constrained neural network architectures work well for vision based tasks, since they force the hidden units to develop feature detectors which are specialized to the portion of the input to which they have connections. Similar architectures were used in the fault detection task described in Chapter 5. The network used for face detection is shown in Figure 6-3.

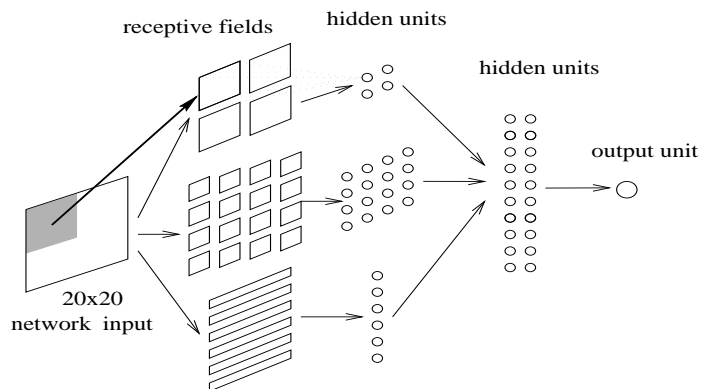


Figure 6-3: The face detection network architecture. Some of the architectures used in this study do not employ the second hidden layer.

Note that it is not the particular configuration of receptive fields that is important in this architecture. Other configurations have also been explored with the same results. The important point of this architecture is that the hidden units in the first hidden layer are only connected to spatially close input units.

Since this architecture is a subset of a fully-connected network, a fully connected network should be able to perform as well. However, a fully-connected network with the same number of hidden units did not perform as well. Perhaps this is because using a fully connected network with the same number of hidden units increases the number of connections by an order of magnitude. Therefore, a fully connected network may require more training examples or larger training times to achieve the same performance.

3. ENCOURAGING DISTRIBUTED INPUT RELIANCE

As described in [Bolt *et al.*, 1991, 1992] [Clay, 1990, 1992] [Judd & Munro, 1993] [Murray & Edwards, 1993] [Sietsma & Dow, 1991], encouraging generalization through noise can be accomplished in many ways. For example, in those studies, noise was introduced into the ANN's weights, into the outputs of the hidden units, and into the input layer(s). However, many of those studies have concentrated on small, synthetic, problems which cannot be extrapolated to real-world vision-based tasks. Further, although introducing noise into the inputs is a good option, in some tasks simple uniform noise may hurt performance, as was shown for the task of autonomous driving [Pomerleau, 1992].

In this study, experiments which introduced noise into two different places in the neural network were conducted. In the first set of experiments, noise was introduced into the outputs of the hidden units. In the second set of experiments, noise was introduced into the input units (this is equivalent to introducing noise into the training examples).

The first set of experiments added noise into the activation of the hidden units. Several different types of noise were used. In these experiments, the hidden units which are connected to the input layer are given a small probability of outputting an incorrect activation. Each hidden unit's probability of being incorrect is independent of all other hidden units. The probability is non-zero only in the training phase; during simulation, the hid-

den units perform normally. For training, three forms of incorrect activations were explored.

1. The first type of noise pinned the outputs of the hidden units to either +1 or -1, depending on whether the correct output is less than 0, or greater than 0, respectively.
2. The second form of incorrect activation was de-activation; the hidden unit's output was set to 0.
3. The third type of incorrect activation set the output of the hidden unit to a random value chosen uniformly from the interval $[-1;+1]$.

By adding noise to the activations of the hidden units, the first two forms of noise (described above) were not beneficial in terms of improving test-set performance. The third form of noise performed well. It provided approximately the same performance as adding noise into the input units, described below. However, there are some complications with the techniques described above. If the network architecture uses hidden units connected to overlapping retinal fields, how should noise be added to each of the corresponding hidden units? If noise is not introduced at the same time to all of the hidden units which are connected to the same input units, the network will develop redundant *similar* feature detectors. This may force the network to pay more attention, in terms of dedicating connections/hidden units, to the easier features. What is desired, however, is redundancy through detecting *different* features. Similar complications arise when the retinal fields only partially overlap. Since adding noise into the input units does not have these complications, and performs as well, it will be explored in detail.

To effectively use noise in the input units for improving task performance, it is necessary to determine both where and how much to noise to add. Before making these decisions, however, it is necessary to analyze where networks trained without noise pay the most attention.

4. ANALYSIS OF NETWORKS TRAINED FOR FACE DETECTION

This section first presents a sensitivity analysis for the networks trained without noise. Second, sensitivity analysis is performed for networks trained with three methods of introducing noise into the input layer.

4.1 Networks Trained Without Noise

To determine the importance of a particular portion of the input retina, a 4x4 square around the portion of the input retina being analyzed is replaced with random noise in all of the examples in the **test** set. After standard training, the network is tested on this modified test set, and the errors are measured at the output layer. This procedure is repeated for all portions of the input retina with the same trained network. As shown in Figure 6-4, when the portions of the inputs which correspond to where the eyes are usually found are replaced with noise, the error increases dramatically. The same is true with the portions of the input which correspond to the nose. Therefore, the network is most sensitive to the portion of the inputs which correspond to the eyes or nose. When the mouth is replaced with noise, the error increase is smaller. The squared error (summed over the entire test set) for the networks trained without noise is 1181 on the test set (averaged over 10 runs). This simple method reveals easily interpretable results for the problems examined in this paper.

Why are the networks less sensitive to the mouth regions? The eyes and nose, unlike the mouth, all appear fairly similar in many sample images; therefore, they are easy features to use. However, the appearance of the mouth is variable across the set of training images; it can be in a variety of poses. For example, a person may or may not smile in a photograph. Even when a person smiles, the mouth may or may not be open; therefore, the person's teeth may or may not be showing. On the other hand, when face examples were gathered for training, most often the eyes were open, and appeared, at the 20x20 resolution, as fairly uniform dark regions in the image, as can be seen in Figure 6-1.

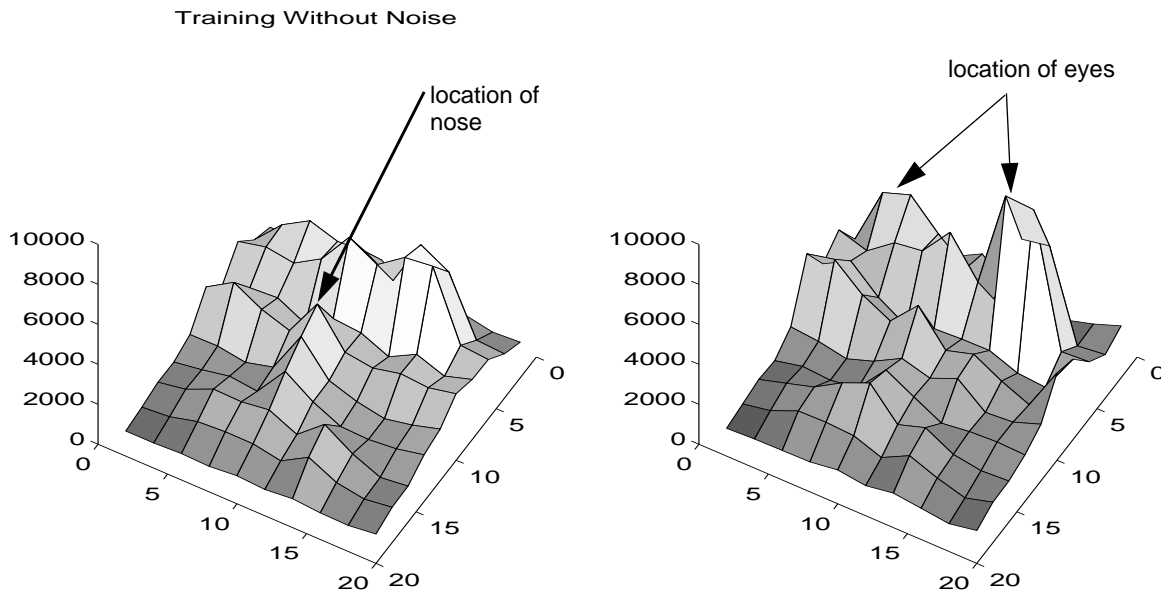


Figure 6-4: Sensitivity analysis for networks trained without noise. **Left:** Average sensitivity of 10 runs. **Right:** Representative sample run. Note that error increases dramatically when the eyes or the nose are replaced with noise. The error increase is much smaller when the mouth is replaced with noise.

4.2 Networks Trained with “Block-Noise”

The networks trained without noise concentrate very heavily on the eyes. This has also been noted empirically in performance tests of the full system described in [Rowley, Baluja & Kanade, 1996]. However, we would like the network to not only focus on the eyes, but also to use other features of the input which may be available. One way of forcing the network to focus on other features is to remove the eyes and nose from the inputs, and force the network to make discriminations based upon the remaining image. Similar ideas of using structured noise, in the domain of autonomous road following, have been explored in [Pomerleau, 1993].

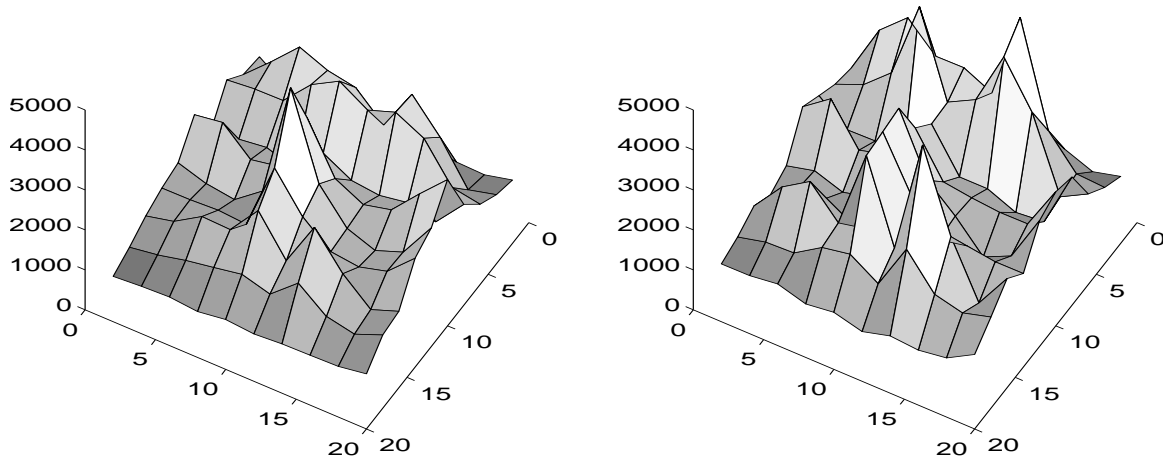
In the set of experiments described in this section, an $N \times N$ square of the input is replaced with uniform noise. This square is positioned randomly, in each example, in the top half of the inputs. Since each example may be presented multiple times to the network during training, the square is positioned in a random position for each pattern presentation. Multiple values of N are tried, since it is not *a priori* known what the best setting is. The

hope is that by replacing the eyes and nose with noise, the network will learn that these are not always reliable features, and will thereby consider other features in the image. The sensitivity of a network trained with 2x2 blocks of noise can be seen in Figure 6-5; there is an increased focus on the nose and mouth. The performance of this network on the test set improves to 1060, an 11% improvement. If too much noise is introduced, for example by using larger blocks (7x7), performance is degraded to 1240. When too much noise is introduced into the networks, features are too frequently obscured during training. Therefore, the network is unable to develop feature detectors for the most important features; the network relies entirely on features in the bottom half of the image. The performance curves, as a function of the block size, are shown in Figure 6-7. The significance of these results is measured using the two-sample Mann-Whitney test. This is a non-parametric equivalent to the pooled two-sample *t*-test. At the 99% confidence level, the difference between the no-noise and 2x2 block size is significant. The difference between the no-noise and the 7x7 block size is significant only to the 87% confidence level.

It is interesting to note that the most prominent features found in the networks trained without noise are also prominent in the networks trained with noise (for small block sizes). The difference between the “noise” and “no-noise” networks is that in the “noise” networks, the secondary features also become more prominent. However, these secondary features change per training session. The randomization of the initial weights plays a larger role in determining which secondary features will be used than in determining which primary features are used, since the primary features are consistent regardless of weight initialization. This can be visually displayed if we look at the average errors of the 10 runs for various noise levels. Because only the primary features are consistently found in all of the runs, the averages look similar (albeit at different scales). One unexpected feature is that the networks start to focus on the cheeks in addition to the mouth. This is due to the fact that the cheeks are also fairly uniform in the face images, and therefore provide a good indicator for the presence of a face.

An interesting test is to apply noise to all portions of the input during training rather than just the upper half of the input retina. In real applications, this would be used when there is little *a priori* knowledge of where the networks are paying attention. The sensitivity for

Training Noise over Eyes (block = 2)



Training Noise over Eyes (block = 7)

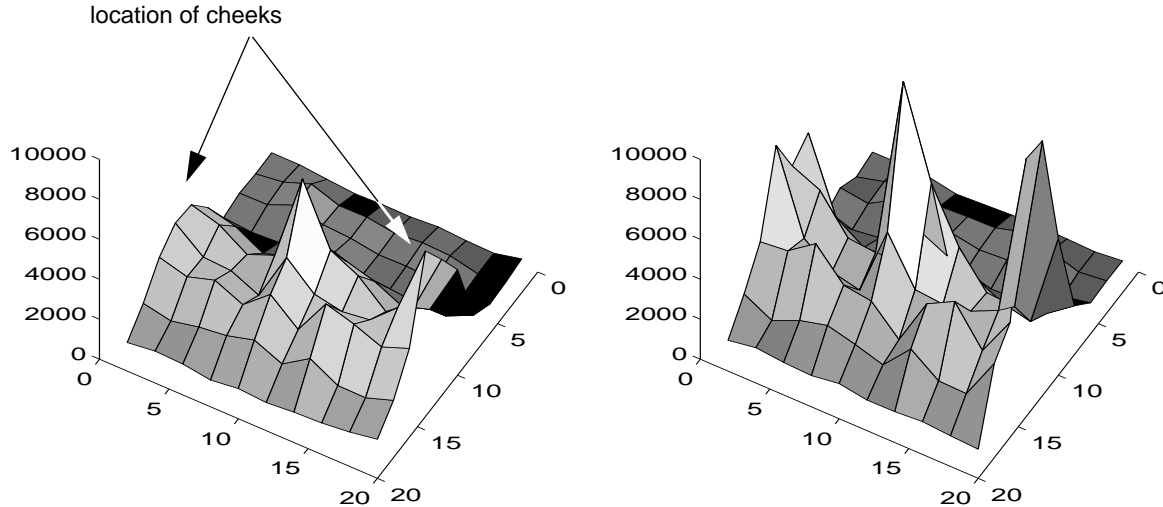


Figure 6-5: Sensitivity to portions of the input image, measured for networks which are trained with noise occurring in a square of size (block) randomly appearing in the top half of the image. Z-axis is errors on test set when the a noise square is placed with its upper left corner in x,y location indicated in the diagram. With block sizes of 2x2, the network still focuses on the eyes and the nose. With too much noise, block sizes of 7x7, the network does not find the important features such as the nose and mouth, and relies entirely on features in the bottom half. **Left:** Average of 10 runs. **Right:** sample run.

the networks trained in this manner are shown in Figure 6-6. As shown with the 2x2 block size, the network is sensitive to all of the features in the input, including eyes, nose and mouth. The error for the 2x2 blocks is 1070. If the block size is increased to 7, as in the case with the noise-over-the-eyes example, the features again become distorted, and the error increases to 1302. Using the Mann-Whitney test at the 99% confidence level, the difference between no-noise and the 2x2 block noise is significant. The difference between the no-noise and the 7x7 block noise is significant at the 98% confidence level.

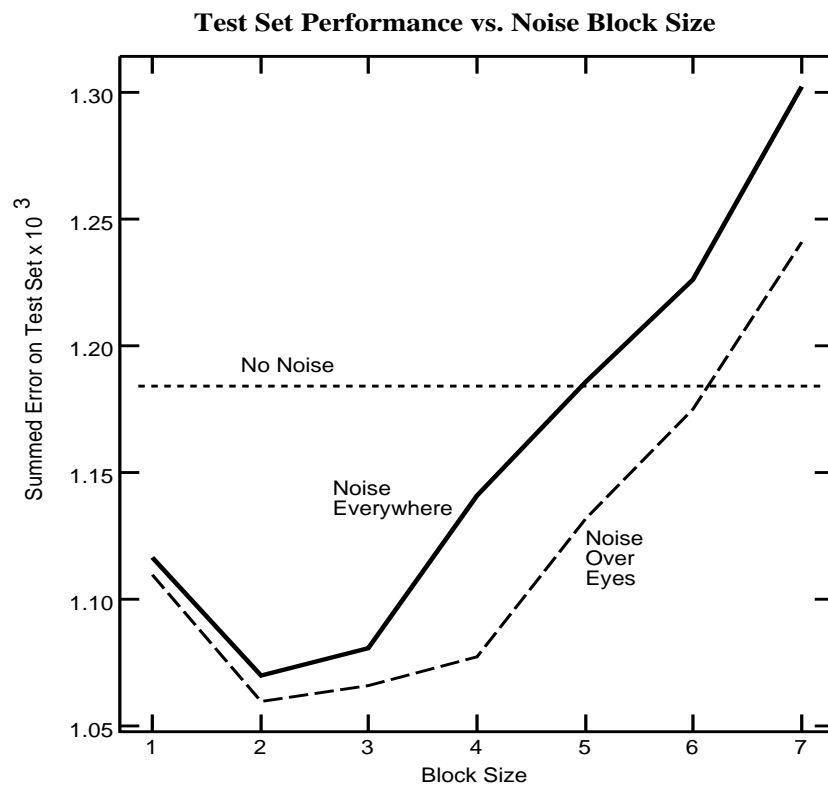
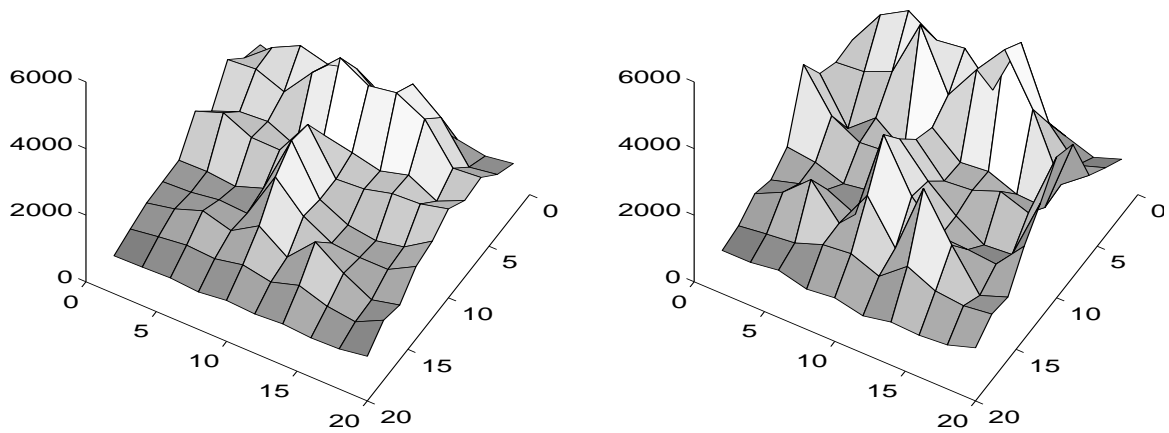


Figure 6-7: Effects of adding noise on test-set performance. Too much noise is detrimental to performance.

4.3 Networks Trained with Uniform Noise

Another technique to add noise to the inputs is to assign an independent probability of noise for each input. Unlike the techniques in the previous section, the noise does not

Training Noise Everywhere (block = 2)



Training Noise Everywhere (block = 7)

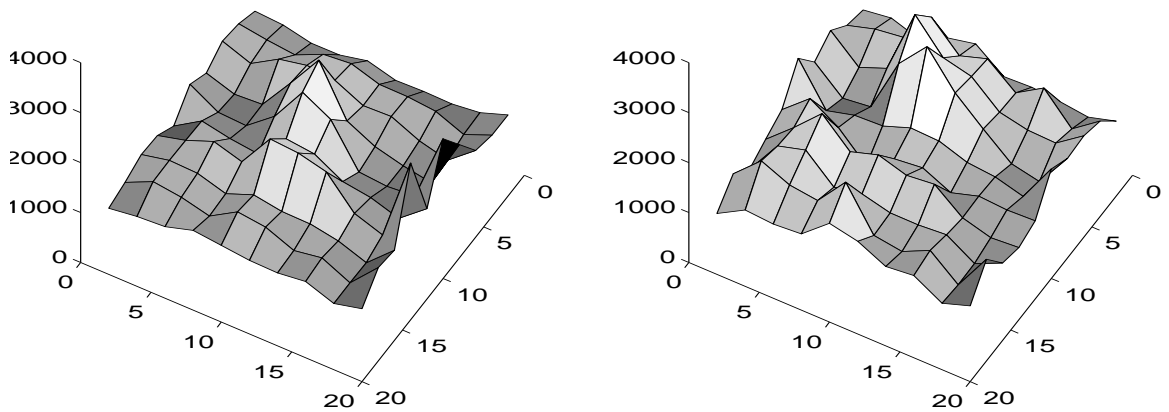


Figure 6-6: Same as Figure 6-5, except that noise is introduced to all regions of the input, not just over the top half of the image. **Left:** Average of 10 runs. **Right:** sample run.

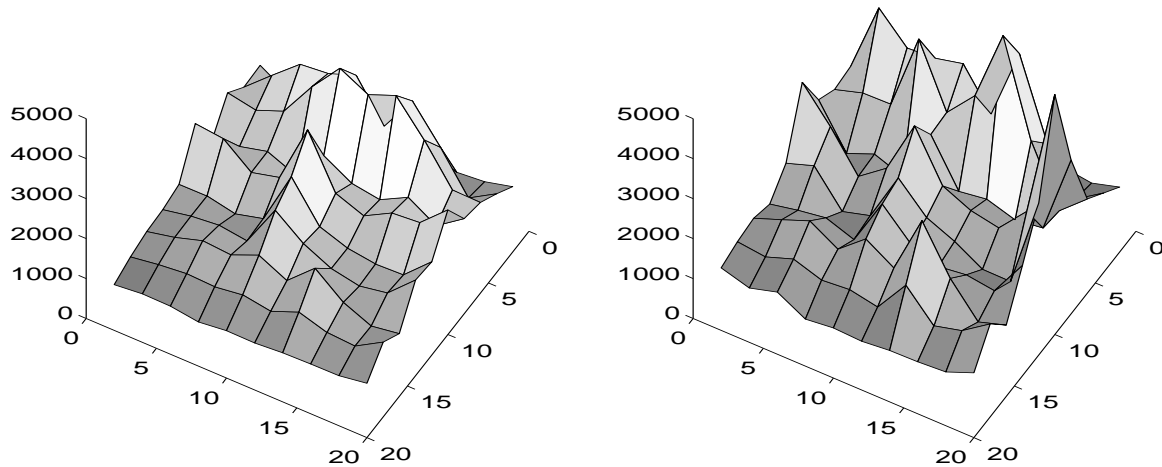
have to appear in blocks, and the number of inputs containing noise in each example is not preset. Various probabilities were examined, ranging from a 1-30% probability for each input. The results are shown in Figure 6-8. Note that networks which are trained with the 2% probability of noise improve in performance, although there seems to be little difference in the portions of the input on which they concentrate. The error on the test set is 1079, a significant improvement over using no noise (at the 97% confidence level). The difference between this error and the error achieved with noise-over-the-eyes(2x2) is not significant. Networks which are trained with the 30% noise are unable to find any usable features, and exhibit an extremely severe degradation in performance. The error on the test set is 7243 (significant to the 99% confidence level). All trials in which greater than 10% noise was introduced resulted in significantly degraded performance over using no-noise at all.

5. SUMMARY AND CONCLUSIONS

This chapter has demonstrated a technique to visualize a network's sensitivity to the inputs. This sensitivity reflects where the network is focusing its attention, and can be thought of as an implicit saliency map. For the task of face detection, methods for modifying the saliency map to encourage the network to attend to more than just the eyes and nose were presented. The resulting networks used the cheeks and mouth in addition to the eyes and nose for detection.

Several methods of introducing noise into the training procedure were explored in this chapter. Since the goal of this chapter was to demonstrate the effects of noise on the implicit focus of attention, and not to select the best method for introducing noise, each of these methods were effective and served the same purpose. By obscuring some of the most easily accessible features, they forced the networks to concentrate on all of the available features. The introduction of noise had a significant impact in performance on an independent test set. A summary of the results obtained by the different methods is provided in Table I.

Training Noise Everywhere (uniform 0.02 prob.)



Training Noise Everywhere (uniform 0.3 prob.)

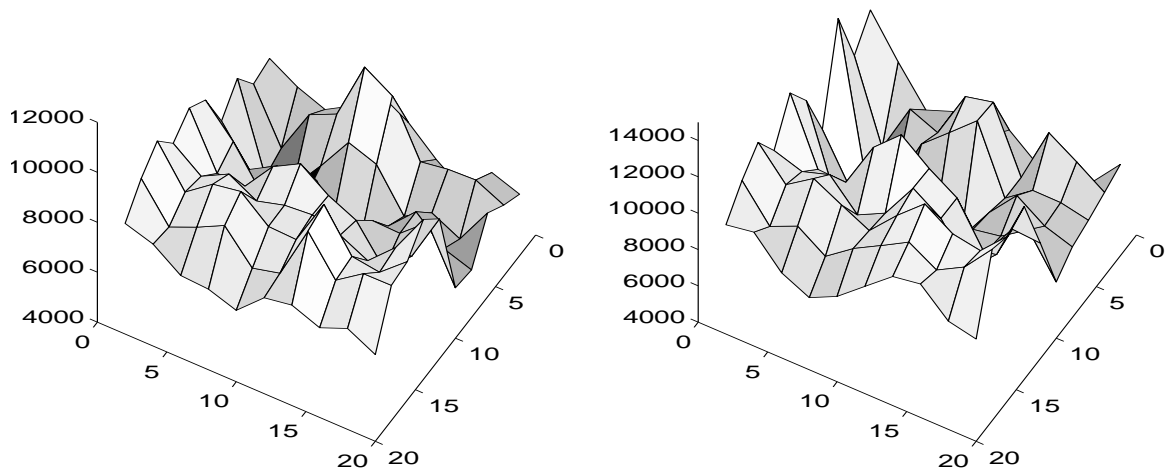


Figure 6-8: Like Figure 6-5, except that the noise added has a uniform probability across all outputs of (top) 1%, and (bottom) 30%. **Left:** Average of 10 runs. **Right:** sample run.

Table I: Effects of Training with Noise: Summary of Results

Method	Amount of Noise	Performance on Test Set (Sum of Squares)
No Noise (Baseline)	—	1181
Block Noise - Only in Top Half of Image	2x2 Block Size	1060
Block Noise - Only in Top Half of Image	7x7 Block Size	1240
Block Noise - Anywhere in Image	2x2 Block Size	1070
Block Noise - Anywhere in Image	7x7 Block Size	1302
Uniform Noise - Each input is independent	(2% probability per input)	1079
Uniform Noise - Each input is independent	(30% probability per input)	7243

One direction for future research is to use these methods of analysis to automatically select or create good training instances. For example, in the face detection domain, if the network is paying too much attention to a particular region of the inputs, such as the eyes, it should be possible to synthesize examples, from the set of labeled training examples, which have that portion of the input obscured. The types of examples presented can be dynamically varied as the network's focus of attention changes. If this technique is to be used, however, the effects of forgetting (i.e. the network becoming more tuned to the current examples while losing what it has learned from the previous ones), will have to be carefully considered.

CHAPTER 7

RELATED WORK

The ideas presented in this thesis draw from a broad range of material in the fields of machine learning and computer vision. In this chapter, the relations between expectation-based selective attention and task transfer, multi-task learning, relevance measures (methods for input selection), and other neural and non-neural selective attention procedures, are discussed. The distinction between covert and overt attention is made with a comparison to active-vision systems. Finally, the chapter ends with some comments on the differences between the approach taken in this thesis and those often used in more traditional object-based vision systems.

1. ALTERNATE NEURAL APPROACHES FOR SELECTIVE ATTENTION

Many different neural approaches to selective attention have been presented [Olshausen & Koch, 1995] [Koch & Ullman, 1985] [Ahmad, 1991] [Mozer, 1988] [Laar et al., 1995], [Schmidhuber, 1991], [Fukushima, 1987], [Olshausen, *et al.*, 1992], [Goddard, 1992], to name a few. Many of these studies, like those often performed in the psychobiological literature, have concentrated on attention in static images. A brief review of some of the major points of the works listed above are given below.

The work of [Koch & Ullman, 1985] and other computational models is reviewed in [Olshausen & Koch, 1995]. One of their basic approaches to visual attention, which many other models have since used, is to direct attention through (possibly multiple) saliency maps. In their model, the saliency map does not code for particular features, like color, but rather for how *different* a particular region is relative to its neighbors. This map may also be thought of as a “conspicuous-ness” map. A winner-take-all (WTA) mechanism then determines the most salient feature and directs attention to it via gating.

The model presented in the previous paragraph, by [Koch & Ullman, 1985], considers inputs which are unlike their neighbors to be important. Other proposed definitions of saliency require the creation of maps based upon the detection of specific features. These features are chosen, and weighted, by the requirements of the task. One such connectionist-based model for studying covert visual attention is termed VISIT [Ahmad, 1991]. Simulations show that the network’s behavior matches much of the known psychophysical data on human visual attention [Ahmad, 1991]. VISIT is composed of four main networks. The two which are most relevant to this discussion are the gating network and the priority network. The gating network is responsible for suppressing all activity except for the region of attention. The priority network determines which regions are to be gated. The priority network can use either bottom-up or top-down information. In VISIT, pre-selected feature-detectors are applied to all areas of the image. For example, in an image which contains red and blue bars which can be oriented vertically or horizontally, four feature maps may be used. One which responds to red color, another to blue color, another to vertical orientation, and one to horizontal orientation. Top-down information is used to select which feature maps are used. When searching for blue, vertical, bars, the

red and horizontal feature maps are not used. The bottom-up information is gathered from the activation of the remaining, activated, feature maps. The location of the activation in these feature maps can be used for focusing attention. Either a single feature map or conjunctions of multiple feature maps can be used. A possibility for extending this system to real images is to use the principal components of the image as the set of features to base the feature-maps [Ahmad, 1991].

With similar feature detectors as described in VISIT, [Mozer, 1988] has presented a model of attention, termed AM (Attentional Mechanism), which is designed to address the problem of having multiple, possibly conflicting, suggestions of where to focus attention next. AM is part of a larger connectionist object recognition system, termed MORSEL, which uses AM to guide processing. The AM model gates the activity of feature maps to higher levels, like the VISIT system, described above. External knowledge sources are again required to provide suggestions about which features should be used, and how to weight the features. An iterative relaxation rule is used to resolve multiple suggestions; at each time step, the units adjust their values according to a locally operating competitive update rule. The updates are based upon the unit's previous activations, as well as the activations of its neighbors. Eventually, the network selects the region of maximal activity. As pointed out in [Ahmad, 1991], one of the problems of an iterative model is that the settling time may be sensitive to not only the size of the image, but also to the contents of the image.

An interesting observation that Mozer makes, which is relevant to the types of attentional mechanisms described in this thesis, is that there is a distinction between data driven and conceptually driven guidance of the spotlight¹. "Data-driven" corresponds to control based on the current input image – for example, drawing attention to objects rather than empty regions. This can be implemented by having inputs from all of the feature maps; this will draw the spotlight to regions in which the features were found. "Conceptually-driven", on the other hand, corresponds to higher level control of the spotlight mechanism. For example, consider the task of reading, in which the subject must scan the text

1. As a reminder: the areas in which the spotlight focuses are operatively defined to be the areas in which an improved performance is found in one or more of the tasks of stimulus detection, identification, localization, or simple and choice response times [Umiltà, 1988]

from left to right. This type of conceptual information is closely related to how expectations are used in this thesis; domain knowledge is used to focus attention over time.

[Laars *et al.*, 1996] present a model of covert attention which, like many of the other methods described here, uses templates or pre-developed features to create feature maps. These feature maps contribute to the overall priority map, which determines which locations of the input that are the most important. However, in this implementation, the features are weighted by an artificial neural network, which uses task-specific inputs to determine the weight for the contribution of each of the feature maps.

[Schmidhuber & Huber, 1991] presented an interesting reinforcement learning approach (see [Kaelbling *et al.*, 1996] for a good review of reinforcement learning techniques) to learning artificial fovea trajectories. Consider the following task: a 2D object, with arbitrary rotation and translation, is located on a pixel plane. Learn to give the position and orientation of a predefined target on the object. In their formulation, the task was defined as a “reward-at-goal” task, in which steps towards the goal do not receive explicit feedback from the environment. Instead of inputting the entire 2D input array to the network, as was done in the methods proposed in this thesis, the network was only given a small localized “receptive field”. The inputs were not pixel based; instead each of the inputs to the network covered a region of pixel inputs. The number of pixel inputs which were covered by network input changed based upon what portion of the fovea the input represented. The inputs which represented regions closer to the center of the fovea were at higher resolutions than those which were further from the center. The task was to move the fovea, which specifies the focus of attention, over the predefined target.

The system developed to address this problem employed two networks. The first network, the “control” network, took as input the receptive field (as described above). The control network’s output was interpreted as a direction to move the receptive field. The second network, the “model” network, used the inputs and outputs of the control network as inputs. The model network was trained to predict what the next inputs to the controller would look like after the controller’s output action were taken. The model network was trained first. It was trained by generating examples by placing the fovea on a random location in the inputs, selecting random actions, and using the resulting fovea

input as the target output for the model network. After training the network, its weights were frozen. The final system was trained by using the outputs of the controller network as hidden units in a bigger network which used the model network as the top layer (see Figure 7-1). By using the “unfolding in time” recurrent backpropagation training method, this system was trained to generate foveal trajectories to move the focus of attention to the desired location. The error fed back to the control network is done through the model network, where the error is the difference between the ideal input (when the fovea is centered around the desired target) and the actual input. A trial consisted of controlling the fovea for some number of predefined steps using the control and model networks. The role of the model network is to provide feedback to the controller, since feedback is not made available by a teacher. Although this appears to be a promising approach, the images on which this system has been tested have been relatively simple.

Other systems have also been proposed in the neural network literature, such as [Fukushima, 1987], [Olshausen, *et al.*, 1992], [Goddard, 1992], to name a few. Many of these systems use one or more of the features found in the models discussed here. Many attempt biological plausibility. Many systems also attempt to characterize attentional shifts in the examination of a single image (many times this is done with lateral inhibition, as found in competitive learning or WTA networks). Finally, many of the systems use feature detectors designed to find features which are *a priori* known to be important. The goal of the work presented in this thesis is very different. Attentional shifts were used to focus attention in *future scenes*, based on their expected content. Further, although it is possible to incorporate *a priori* information, as in the hand tracking task presented in Chapter 4, the central focus of this thesis was learning the feature maps, in the absence of *a priori* knowledge.

2. TASK TRANSFER

Learning from multiple tasks as well as learning from other learners are two areas of current interest in the artificial intelligence and machine learning communities. Many similar ideas have been explored in this thesis, as described in the remainder of this section.

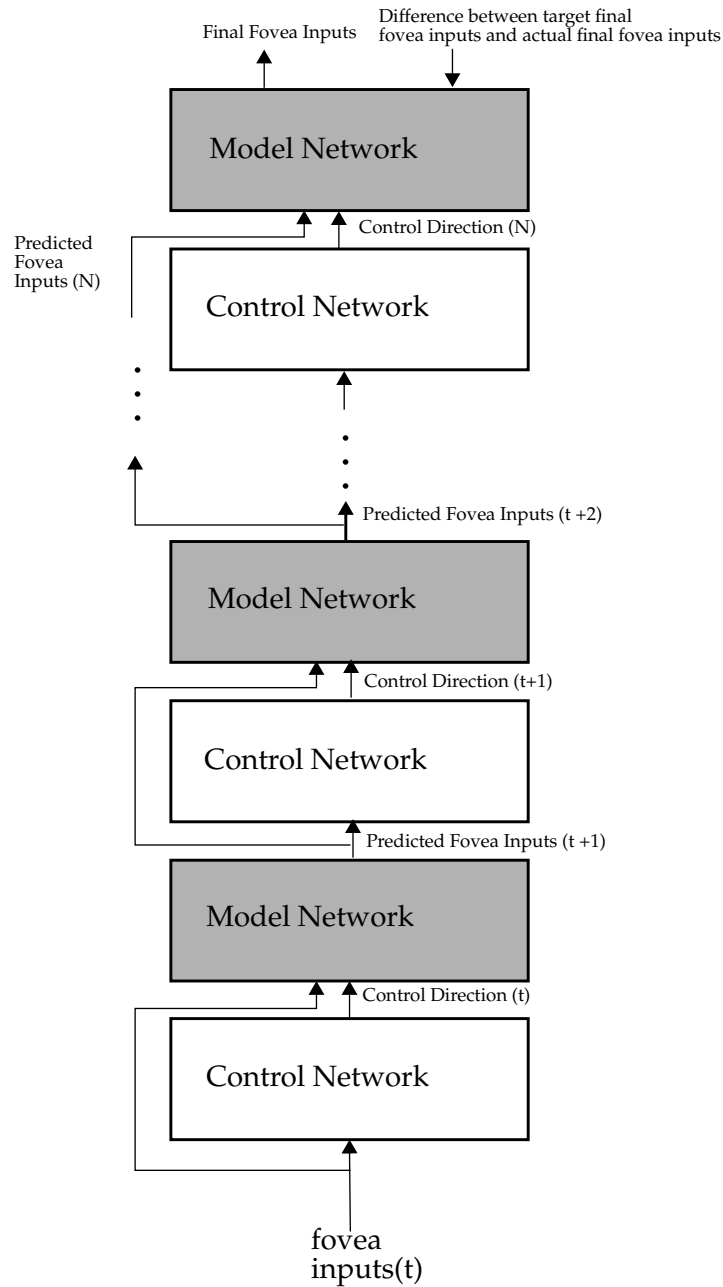


Figure 7-1: One method of using the architecture proposed in [Schmidhuber & Huber, 1991]. The model network's weights are frozen. The model network is used to convert the error in pixels to a derivative for the controller network. The system is simulated for N steps.

2.1 Multitask Learning

The methods described in this thesis are related to *Multitask Learning* (MTL). MTL is a method explored by [Caruana, 1993] for training a network to perform multiple tasks simultaneously, even when there is only a single task of importance. For example, in [Caruana, Baluja & Mitchell, 1996] a network which was trained to predict pneumonia severity given only a few pre-hospital-admittance lab tests was simultaneously trained to predict the outcomes of 35 more expensive lab tests which could be taken in the future (see Figure 7-2). The results on the task of pneumonia severity prediction improved by training the network on multiple tasks over training a network solely on the pneumonia task. In that study, a variant of backpropagation specialized for determining relative ranks, termed *rankprop*, was used. Nonetheless, backpropagation, or any other ANN learning algorithm, could have been employed.

The basic premise of MTL is that since the network is given related tasks to learn, some of the features which are beneficial for one task may also be useful in solving other tasks. By using the architecture shown in Figure 7-2, the network forces the tasks to share the same hidden layer. Even if the extra tasks do not perform well, they provided a helpful inductive bias for the network trying to learn the main task. For example, in the pneumonia severity prediction task, the predictions of the extra lab tests never trained accurately. However, the bias they provided for the task of interest resulted in statistically significant improvement in the performance of the main task – pneumonia risk assessment.

The related tasks for MTL must be chosen carefully. For example, it is possible to select a set of tasks which are not related and would not benefit from shared internal representations. Simultaneous training of these tasks may hinder performance rather than help it. The tasks may compete for hidden units, or representations, in the hidden layer. In response to this possibility, it is common to use hidden layers which are larger than those commonly used in single task learning experiments [Caruana, 1993].

The experiments described in this thesis utilize many of the same principles as MTL. For example, in the synthetic ‘+’ and ‘x’ discrimination and localization experiments described in Chapter 2: Section 3, there are two tasks which compete for internal repre-

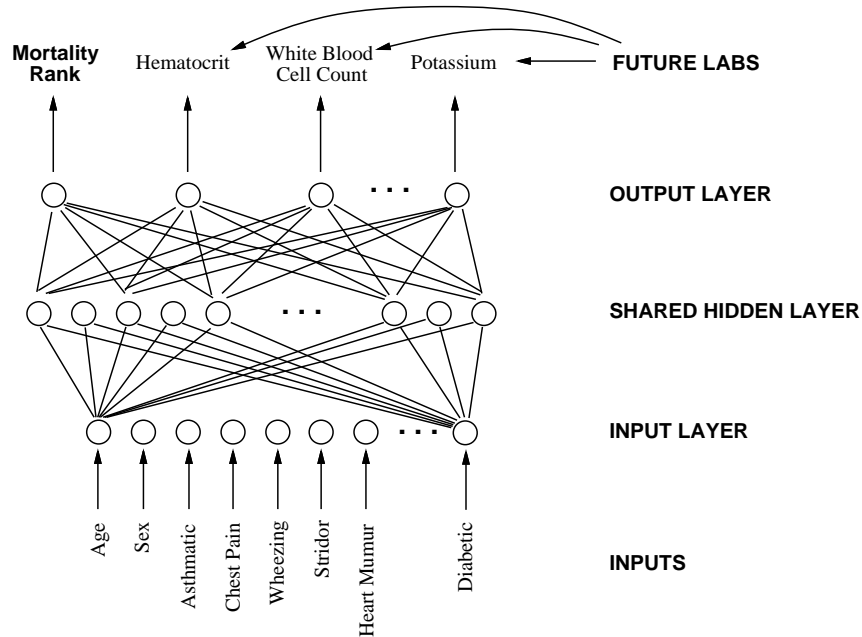


Figure 7-2: Multitask learning for relative pneumonia severity prediction. The network is a fully connected feed-forward network. Figure adapted from [Caruana, Baluja & Mitchell, 1995].

sentations: the location task and the discrimination task. As a reminder, note that the prediction task does *not* compete for internal representations because of the “forward-only” connections. (As defined earlier, these connections ensure that the hidden units do not encode prediction information by not propagating error back from the prediction outputs). In those experiments, changing the internal representations, by forcing the network to encode information from multiple tasks, has a direct effect on the ability of the network to make predictions of the next state.

2.2 Learning from Other Learners

Another method of exploiting learned domain knowledge has been described in [Mitchell, O’Sullivan & Thrun, 1994] [Mitchell & Thrun, 1993], in which *Explanation-Based Neural Network Learning* (EBNN) was explored. This approach is closely related to the methods explored by [Schmidhuber & Huber, 1991] (described previously). In EBNN, models of the environment, which have been previously learned by separate neural networks, are

used to help teach new networks more complex tasks. For example, consider a robotics task of finding doorways. The previously learned knowledge may include (1) detecting doorways when the robot is next to them and (2) predicting what the inputs to the network will be if the robot takes an action of moving 1 meter from its current position. With this domain knowledge, it should be possible to help another system learn to detect doorways 1 meter ahead. Further, by chaining multiple copies of the second model together, doorways which are 2 or more meters ahead should also be able to be detected. The key to this method lies in the training algorithm used, *TangentProp* [Simard *et al.*, 1992]. *TangentProp* adjusts the network weights to reduce the error in not only the values of the outputs, but also in the derivatives of the function represented by the network. The inputs which are most meaningful are determined by using the model networks to *explain their actions*. The weights and activations of the model networks are examined in order to analytically derive the partial derivative of the network's output with respect to each input feature. *TangentProp* is used to ensure that the derivatives are matched in the target network also.

In this thesis, the distinction between domain knowledge and the receiver of this knowledge is much less precise than in EBNN. Unlike in EBNN and other studies which have examined task transfer, the prediction is used for preprocessing the inputs, rather than explicitly training a new learner. Even during simulation, when the models are used, there is no distinction between the task-transfer learning phase and the simulation phase. For the architectures examined in Chapter 2, the filtering mechanisms not only affect the classification/regression network, but also affect the model network. As the filtering gets better, the important features are more easily found, thus improving the filtering. This provides a means for the two learners to bootstrap from each other's learning.

3. LINK-PREDICTIVE NEURAL NETWORKS

[Tebelskis, 1995] explored the use of predictive networks as acoustic models for large vocabulary recognition of isolated words and continuous speech. In his model, a network took as input several frames of speech fragments and predicted the next frame. The predicted frame is compared to the actual next frame. If the error is small, the network is

considered a good model of the speech segment. Separate networks were used to model each phoneme of interest. The classification of the speech segment is assumed to be the phoneme associated with the network which created the prediction with the lowest error.

This procedure is somewhat similar to those used in this thesis. Tebelskis, however, compared the prediction directly with the next inputs. He used the magnitude of the difference between the predicted and actual frame to determine the correct classification. This method may not work well if many of the inputs are not predictable – for example if some inputs usually contain noise. If these types of inputs exist, just using the difference may yield inconsistent results. In this thesis, because of the inherent uncertainty in making predictions, the prediction is always used as a filter for the original inputs or as extra inputs to a higher-level learning system. These ideas can be extended to Tebelskis’s work as follows. In addition to the prediction network Tebelskis used, a separate “comparison” network which takes as input the predicted and actual frame, can be trained to output whether the two belong to same class. One of these networks can be used for each phoneme examined. This will allow the network to concentrate on inputs that can reliably be predicted and ignore those which cannot. Therefore, even if the prediction is inaccurate in some of the irrelevant inputs, if it is accurate in the relevant ones, a correct classification can be made. A problem may arise when multiple “comparison” networks respond positively to the frame. In this case, the sum over multiple frames of the same speech fragment, or, as was originally done by Tebelskis, the Euclidean distance between the predicted and actual frame can be used to break ties.

4. INTEGRATING SALIENCY OVER TIME - INPUT RELEVANCE

In many real world tasks, such as anomaly detection and factory monitoring, it is often the case that there are many inputs to a learning system which are not relevant. For example, consider the semiconductor plasma-etch fault detection system which was discussed in Chapter 5. For anomaly detection, this study only used the wavelength of light emitted at the 520nm wavelength. However, there are over 400 sensors in the plasma etch chamber. It is an extremely difficult task to find which of those sensors provide the most information. Although sensors are becoming increasingly available, the technologies required

to handle all of the information they provide, which is often stored in real-time, is lacking. Therefore, there is a growing interest in the study of the *relevance* of inputs [Caruana & Freitag, 1994][John *et al.*, 1994][Langley & Sage, 1994][Mozer & Smolensky, 1989][Has-sibi, *et al.*, 1994][Kohavi & John, 1995][Alumualim & Dietterich, 1991]. For example, if some inputs are not relevant to the task, most learners (neural networks, decision trees, etc.) perform better if these inputs are removed before learning is initiated. Removal of these inputs reduces the dimensionality of the inputs and also reduces the potential for finding spurious correlations given the same amount of training data.

Vision based systems also have the problem described above, in which there are many inputs, but only a few of them may be important. However, the problems in vision are compounded by changing relevance. Studies of relevance have not addressed the notion of changing relevance. In previous studies, relevance is used to remove inputs that are deemed to *never* be important.

Traditional methods of relevance assessment will not work well in many of the domains described in this thesis. For example, recall the cross detection tasks explored in Chapter 3, Section 2. In those experiments, a cross appeared in any of 16 locations in the input, in a predefined order. Therefore, any of the 16 portions of that input would be considered equally relevant, when examined over a large number of examples. However, at any particular instant, only one of the locations is important. The fundamental difference between the form of relevance needed for vision based tasks, and the relevance measures which have previously been explored, is that a notion of time is necessary. The methods explored in this thesis provide a method for determining *dynamic relevance*. Although inputs are never removed because they may at some time be important, they are de-emphasized because they are not currently important.

Given a method to ascertain dynamic relevance, it should be possible to integrate the saliency over time, and to determine a measure of static salience, or salience irrespective of time. For example, suppose that instead of allowing the markers in the previous experiments to move to all 16 locations, that they were allowed to move to only 12 of the locations. In this scenario, 4 of the locations should be determined to be irrelevant through all time. As a second example, consider tracking a moving object. If the object never enters a

portion of the input retina, then it should be possible to eliminate those inputs from processing. To ascertain time independent relevancy, the relevance of each input, as determined by the methods described to this point, is integrated over time. This is implemented as follows. For each input, a counter is maintained. The counter adds the saliency map's activation for that input at each time step. At the end of training, the summed totals will give an indication of each input's relative importance. An example to demonstrate this point is provided below.

To graphically display the results, a very simple, synthetic, task is attempted. It should be emphasized that this approach can easily be extended to the real-world tasks explored throughout the thesis. This task presents a version of the cross localization problem discussed in Chapter 3. In this task, there are only four locations in which the cross can appear. The system cycles through these locations in a pre-specified order. Like the previous cross location task, distracting crosses can appear randomly in any of the four locations; the system must be able to ignore these extra crosses. In addition, the rest of the inputs may also contain noise. In this task, the best response for non-temporal relevancy is to define all four crosses to be relevant, while defining all other locations as irrelevant. This will give the network the most resistance to confusion by extra crosses and the random noise. For this task, the input retina size is 100 units. The crosses, which appear in non-overlapping locations, cover an area of 5 pixels each. Therefore, in this task, there are 100 inputs, out of which only 20 contain potentially relevant information. Sample inputs and target outputs are shown in Figure 7-3; note the large amounts of noise in the inputs. Relevance information, gathered from the integration described above, is shown in Figure 7-4. The described method is able to find all of the relevant inputs, without any extra training or extra information.

In the studies presented in each chapter (except for Chapter 6), the data was temporally related. Previously, relevance was determined with respect to time, in terms of focusing attention, and de-emphasizing the irrelevant inputs. In this section, a method to measure *time-independent* relevance was given, which can be used to prune away the irrelevant inputs. Most previous relevance studies have examined data which is not temporally related. In addition to the methods described in Chapter 6, the methods used here can

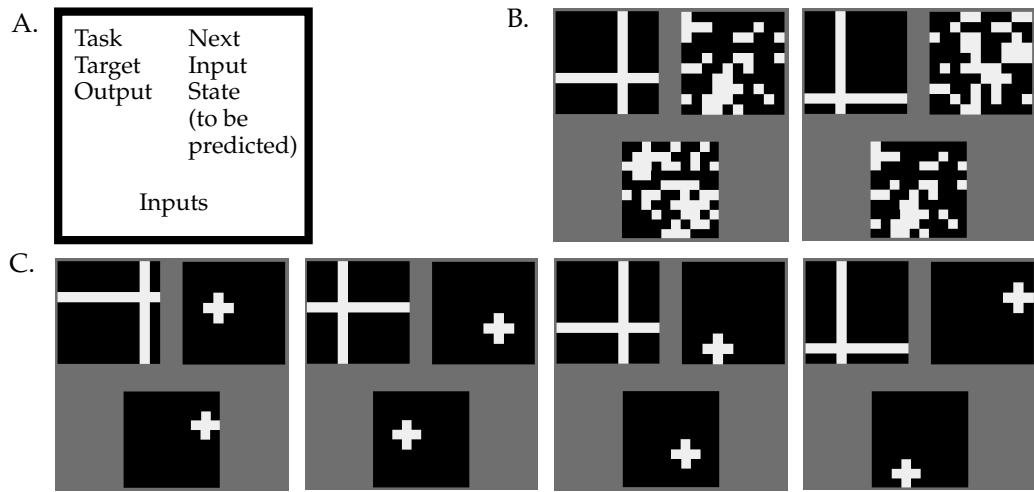


Figure 7-3: A. Legend. B. Actual inputs to the system. Notice the large amount of noise in the inputs. C. Task shown without any distracting noise (these frames were not used in any experiments). Frames in B & C are aligned with each other (top to bottom) to display where the network should be paying attention, and the next relevant locations.

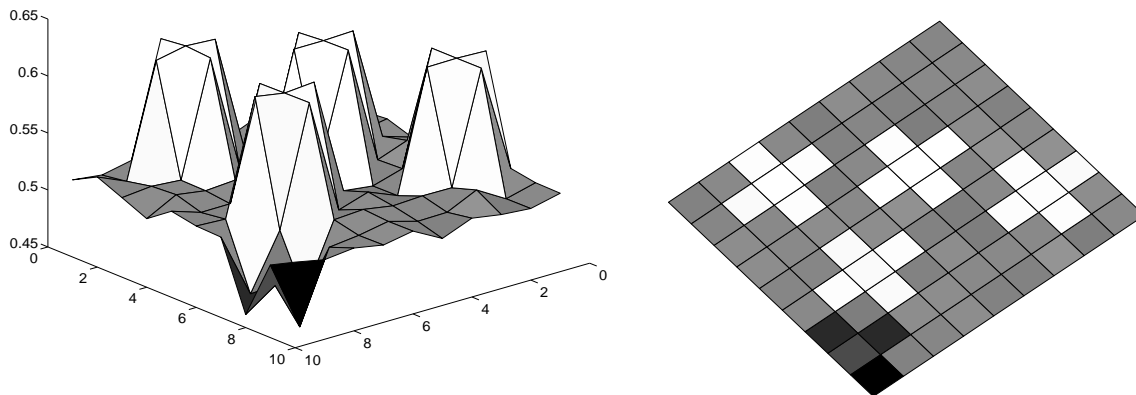


Figure 7-4: Two views of the pixels which were found relevant. Intensity in both views indicates relevance. This is the summed values, over time, of the saliency map for each pixel (normalized). Note that these views are rotated from those in Figure 7-3 for viewing

easily be extended to these tasks by returning to the original modification of IRRE, described in the Chapter 2. By using the extra output layer to *reconstruct the current input* (instead of using it to predict the next input), the relevant features in the inputs can be

determined. By measuring how well each input is reconstructed, and integrating over all examples, an estimate of which inputs are not relevant can be obtained. From this estimate, these inputs can be removed, and the network retrained or the training continued.

One limitation of the approaches described in this section is that they will not eliminate inputs which contain constant values, as these can always be predicted/reconstructed easily. However, as described in Chapter 2, these attributes can be trivially eliminated through pre-processing.

5. KALMAN FILTERS

A form of filtering which is often used in robotics and vision domains is Kalman filters. A good overview of Kalman filters is provided in [Maybeck, 1990]:

[Kalman Filters process] all available measurements, regardless of their precision, to estimate the current value of the variables of interest, with use of (1) knowledge of the system and measurement device dynamics, (2) the statistical description of the system noises, measurement errors, and uncertainty in the dynamics models, and (3) any available information about initial conditions of the variables of interest. For example, to determine the velocity of an aircraft, one could use a Doppler radar, or the velocity indications of an inertial navigation system, or the pitot and static pressure and relative wind information in the air data system. Rather than ignore any of these outputs, a Kalman filter could be built to combine all this data and knowledge of the various system dynamics to generate an overall best estimate of velocity [Maybeck, 1990, pp. 194].

Kalman Filters are used to combine multiple observations from the same sensor as well as observations from multiple sensors to estimate a set of parameters called the “state”. Kalman filters are optimal methods of combining information when the system can be described through a linear model and the system and measurement noises are white (not correlated in time) and Gaussian in amplitude.

Three uses of Kalman filters are discussed here, in the context of the hand-tracking experiments presented in Chapter 4:

1. The location of the hand in each image is the observation and state.

2. The image is the observation and the hand location/velocity, etc., is the state.
3. The image is the observation and the state.

In the first case, in which the position of the hand in each image is the observation and state, the Kalman Filter (KF) can easily be combined with the system presented in Chapter 4. The KF can provide a good estimate of where the hand will be in the next time step, given previous locations of the hand. The estimates of the location of the hand are obtained by a neural network. The NN examines the input image and outputs the hand location. To use this method, however, a noise model for the NN outputs will be required. The NN is not needed for creating predictions of where the hand will be in the next frame, since previous estimates are combined and used for estimates by KFs.

Additionally, it should be noted that the prediction from the KF can be used by employing similar filtering methods to those used in Chapter 4, by using the predictions as a binary filter to remove the irrelevant portions of the image. It should be noted, however, that this makes the measurements dependent on each other, since the prediction of the next hand location is used to filter the image before the NN uses it as input.

The predictions of the KF can also be used in a similar manner as the filtering methods described in Chapter 2. From the hand-location information, an image which contains the hand in the predicted location can be constructed. The actual input image can be filtered based on its differences from the constructed image. The final image is used as input to the NN.

In the second case, the observation is the image, and the state is the hand location and velocity. This case may be difficult to use since there may not exist a linear relationship between the image and the location of the hand. This is exactly the role the neural network plays - mapping image to location. However, it should be noted that the KF has the advantage of using multiple-previous time-states, which were not used in the experiments presented in Chapter 4.

In the third case, the image is the observation and the state. Here, the KF is used to directly make the image predictions. The KF uses linear combinations of the previous images to create the predicted image. This predicted image is linearly combined with the observed image. The idea of predicting images is similar to the techniques explored in Chapter 2. However, more complex combinations (than linear combinations) of prediction and actual images are employed in this thesis. For example, in Chapter 2, prediction and actual images were filtered, pixel-by-pixel, based upon their differences. The larger the difference in pixel intensity, the greater the attenuation of that pixel. This de-emphasized the unexpected pixels. In Chapter 4, images were masked by binary filters to remove irrelevant portions of the image. Finally, in Chapter 5, both the prediction and actual image were used as input to another neural network, which can form arbitrarily complex filtering strategies.

In all of these models, it should be noted that the systems modeled with the techniques explored in this thesis do not have to be linear. Although the original Kalman filter was designed for linear systems, extensions to standard Kalman filters can also handle some nonlinearities.

6. ACTIVE VISION

In contrast to the *covert* form of attention that has been studied in this thesis, *overt* attention has been studied in the active vision communities. Active vision systems have mechanisms that can actively control camera parameters, such as orientation, focus, zoom, etc., to the requirements of external stimuli [Swain & Stricker, 1993][Ballard & Ozcandarli, 1988]. A review of some active vision systems can be found in [Swain, 1993][Swain, 1994][Blake & Yuille, 1992]. An application to face-tracking can be found in [Darrell *et al.*, 1996]. See [Tsotsos, 1992][Aloimonos *et. al*, 1988] for a more theoretical basis. An interesting related topic is that of active perception, where what is perceived is not limited to vision. An application of active perception for a planetary rover project can be found in [Krotkov, 1992].

The goal of active vision systems is, like the goal of covert attention systems, to simplify

the computations of early vision. Active vision systems emphasize the ability to examine areas of interest with high resolution, without examining every region with the same uniformly high resolution. The active vision system can be viewed as a method for selective data reduction. For example, foveated sensors allow the most processing to be focused only on small regions of the input. The covert attention systems described in this thesis also perform the same function, but in a different manner. Here, the inputs which are not of interest are de-emphasized to a background intensity value, so that the regions of interest can be emphasized.

The core of the active vision system is deciding where to move the camera to focus next. One of the difficulties that active vision systems will face, when more complex tasks or multiple tasks are undertaken, is that of *competing tasks*. Suppose that there is only a single camera for solving multiple tasks. For example, consider the Navlab vehicles, in which in one mode of operation, obstacle avoidance and road following are done from the same camera input image. There is the potential problem of deciding where to focus to allow enough filtering for both tasks. In the covert attention models, such as presented here, the filtering is done without changing the camera parameters, rather it is done by filtering the image. However, the active vision system will have the advantage of being able to increase resolution in the areas which are interesting. The covert attention systems described here cannot do this.

One learning approach to active vision was explored by [Rimey & Brown, 1992a and 1992b]. In this approach, multiple Bayes-nets were used to guide which operators to use and to which regions of the image these operators should be applied. A *task-network*, which contains domain knowledge about the desired tasks, is used to divide larger tasks into smaller subtasks. An *expected-area-network* tells where object **X** is expected to be, if the position of another object **Y** is known. When searching for multiple related objects, this network is used to narrow search regions given the placements of previously found objects. The example task given in their paper is visually determining whether a table is set for a formal or informal meal by determining the types of objects which are on table. Following their example, given the location of a knife in the input scene, the expected-area network will narrow the places to search for a fork. If a plate is found as well, this

can further constrain the search. The goal, however, is not to find all of the objects since not all of the objects need to be found. Attention is focused based upon the task. The task-network tells which objects are necessary to be found and the expected-area-network tells the likely locations to find the objects. One of the features of this system is that the utility of carrying out an action can be explicitly derived. For example, if the cost for applying a search operator to a certain size region in the image is known, both the size of the region which must be examined and the expected benefit of finding an object can be considered before selecting the next action to take. Alternate approaches to attention which use Bayes-Networks, although with covert attention, can be found in [Howarth & Buxton, 1993].

Other approaches to active vision are very closely related to the neural network model described in [Ahmad, 1991]. In these models, “primitives” are first extracted in parallel across the entire visual field. Simple primitives such as line orientation or color, as well as more complex primitives such as detecting regions that are different from their surroundings, can be used. The result of this parallel extraction is a feature map which indicates the presence of specific features in each location in the image. The results from these feature maps are combined by assigning different weights to each of the features. This yields the saliency map, which indicates where to move the camera for focusing attention. For an example of such a system, see [Clark & Ferrier, 1992].

7. DICKMANNS’ 4D APPROACH TO DYNAMIC VISION

The work by Dickmanns [Dickmanns, 1992] is closely related to the work in this thesis. Dickmanns’ and his colleagues have studied the control of an autonomous vehicle with active vision. The underlying premise of their work has been in using prediction of the future state to help guide attention, for example, by controlling the direction of a camera to acquire accurate position of landmarks.

Very strong models of the vehicle motion, the appearance of objects of interest (such as the road, road-signs, and other vehicles), and the motion of these objects are encoded in the system. The motion of the vehicle and the other objects in the scene are modeled inde-

pendently. Their system does not use learning. Initially, low level pixel processing is used to find where objects in the scene are located. If all of the objects in the scene are found and good models are available, a forward projection can be made of the scene in the next time steps. By examining the differences between the prediction and the actual scene, the models can be refined. Many of the models used by Dickmanns are based on extended Kalman Filters.

The work by Dickmanns and the work presented here differs in several key aspects. The first is in the amount of *a priori* knowledge that is used. In Dickmanns' approach, a tremendous amount of problem specific information is incorporated into creating the models which are used for prediction and control. In the approach presented in this thesis, the main object is to automatically learn, from examples, the features which should be attended; therefore, very little knowledge about the domain is assumed. Learning is used for detecting the important features, developing the control strategy from the detected features, and also generating the predictive model. Second, Dickmanns' uses all of the previous time steps to make predictions. By using recursive estimation procedures, such as Kalman Filters, information from all of the previous images is combined to create predictions. In the experiments conducted in this thesis, only the single previous image was used for prediction (although this can easily be extended, as described earlier). Finally, the work by Dickmanns' attempts to explicitly model the transitions of the objects in the road with respect to the vehicle. The study presented in Chapter 2 did not address the problem of obstacle avoidance or object detection; only road following was considered. However, a simple extension of this work for obstacle detection is to compare the prediction of the road with the actual image, and to emphasize the differences. This would highlight potential obstacles. However, this extension does not track the obstacles; instead, it treats each frame independently. To track the obstacles, like Dickmanns' approach, a separate procedure must be used.

In designing a real-world application for public deployment, all of the available problem-specific knowledge should be used. Nonetheless, the integration of learning into the design of Dickmanns' system may reduce the amount of hand-coded knowledge which must be acquired and integrated. Further, using learning/adaptation in execution may

increase the reliability of the system in unanticipated conditions – such as changing/adverse weather conditions or periods of sudden lighting changes, such as entering a tunnel [Pomerleau, 1995].

8. AN OBJECT-BASED APPROACH

An alternate approach to the problems examined in this thesis, in particular the lane-marking detection task, is to understand the contents of the input scene, for example, by incorporating object detection and recognition. This approach first attempts to transform the input image into symbols, from which a set of rules can be matched, to direct filtering procedures for future frames. This approach relies upon the ability to segment the input scene into the objects of which it is composed.

In contrast to the above methods, the techniques presented in this thesis are not necessarily *object* oriented. Although the neural network may detect objects, objects in the visual scene do not have to be *explicitly* recognized/detected. For example in the lane-marking detection task, passing cars, trees in the periphery, etc., do not need to be recognized, they only need to be interpreted as intensities which were not expected. Filtering is done on a pixel-by-pixel basis, rather than by (de-)emphasizing full objects. Further, the lane-marking does not even need to be recognized, rather its intensities yield information about the correct output. In terms of how a human understands a scene, there is very little “scene understanding” incorporated into the system. The inputs are mapped directly into the outputs, without decomposing the pixel-inputs into usable symbols.

Some of the interesting points of each of these approaches are listed below. Many of the listed points are fairly general, and also arise in the discussion of symbolic vs. sub-symbolic computation methods. The more traditional artificial intelligence approach, in which symbols are extracted, has the following advantages:

- Intermediate results are more interpretable. The features/objects which are found can be enumerated and easily located in the image.
- The rules for focusing attention can be changed without changing the recognition portion of the system. Extracting symbols can be independent of how the symbols are used.

- Symbols can be used for many tasks, including predicting the next scene.
- Does not rely on being able to predict the activations of individual inputs/pixels.

The approach presented in this thesis has the following advantages:

- It does not rely on general object detection methods, which are themselves very difficult and remain, for the general case, unsolved.
- The relevant features on which to focus attention can automatically be determined, as described throughout this thesis.
- An internal model of task-specific features is developed. Although this is *not* easily understandable, it is usable for other tasks, as shown in this thesis, such as prediction.

The distinction between pixel-based filtering and object-based filtering will be expanded upon in the next chapter.

CHAPTER 8

CONCLUSIONS AND FUTURE DIRECTIONS

This chapter presents a review and comparison of the different techniques explored throughout this thesis. Also presented are several directions for future research, including new applications for expectation-based selective attention.

1. SUMMARY & CONTRIBUTIONS

In many real-world tasks, the ability to focus attention on the relevant portions of the input is crucial for good performance. This thesis has shown that for temporally coherent inputs, a computed expectation of the next time step's inputs provides a basis upon which to focus attention. Expectations are useful in tasks which arise in visual and non-visual domains, ranging from scene analysis to anomaly detection. Although a variety of methods and architectures have been used in this thesis, all of them have presented ways to modify a saliency map to control where a network focuses its attention. In this section, a brief review and comparison of the different techniques explored throughout this thesis are presented.

1.1 Creating Expectations and Expectation-Based Saliency Maps

When temporally related inputs are available, an expectation of the next input's contents can be constructed based upon the current inputs. A saliency map, which is based upon a comparison between the actual inputs and the expected inputs, indicates which inputs will be important for the task in the next time step. Several methods for computing the saliency map and the expectations of the next time step's inputs were presented in this thesis. They are reviewed here.

The first method, presented in Chapter 2, was in the context of lane-marker tracking for autonomous vehicle control and monitoring. The predictions of the next input were created from the representations in the hidden layer of a network trained to perform the main task. The saliency map indicated which regions of the inputs matched the computed expectations. The activation of the pixels which did not match their expected value were attenuated. The attenuation was proportional to the magnitude of the difference between the expected and actual values. This form of filtering was useful because the location of the important features was predictable over time. In this case, the location of the lane-marker in successive frames could be estimated based upon the previous frames. The saliency map de-emphasizes distractions which are not predictable. Because of the task-specific training of the predictions, even distractions which *are* predictable are de-emphasized. For example, in Chapter 2:Figure 11C, a car was not predicted in the image

although it was present in the previous image and its location in the next image was predictable. Since the task is to find the lane-marker, only information about locating the lane-marker is encoded in the network. Information about other features in the input, even when they are predictable, is not encoded. Since knowledge of the car's movements is not modeled by the network, it cannot be successfully predicted; therefore, it is filtered out.

The second method of creating the saliency map was explored in the hand-tracking task described in Chapter 4. Unlike the method described in Chapter 2, the saliency map was not computed based upon the activations of the differences between the expected input image and the actual input image. Instead of predicting the next image, the location of the hand in the next image was explicitly predicted using hand-coded rules and the estimated location of the hand in the current image. The predicted location was used as the basis of a binary pass-filter for the next image. In terms of the saliency map, the pixels within the window were considered salient, while those outside were not. The activations of the inputs within the window were not attenuated. All of the inputs outside of the window were attenuated. This removed many spurious features which could have confused the network.

A third method of employing expectations, described in Chapter 5, is to use them as extra inputs to the network. This method was employed in the task of anomaly detection in the plasma-etch step of semiconductor wafer fabrication. For this task, the expectations were created using a separate neural network which was trained only to make predictions of the next input. A separate network was used because the hidden units of the detection network contained only information about anomalies. However, information about the regularities is important for the prediction task. See Chapter 5: Section 4.4.1 for more details. Unlike the tasks presented in Chapter 2-4, the *unexpected* features in the input were important; therefore, the unexpected features need to be emphasized. In some anomaly detection tasks, it may be sufficient to use the filtering procedures described earlier to de-emphasize the *expected* features. However, in this task, these methods were not appropriate. De-emphasizing or removing portions of the input would yield poor results since anomaly detection in this task relies upon the relative magnitudes of many inputs.

De-emphasizing/removing only a few of the inputs may create misleading artifacts in the input. Therefore, instead of modifying the original inputs, the expectations were used as extra inputs to the network. Additionally, the input-by-input difference vector between the expected and actual inputs, which is the saliency map, was also used as input. One potential difficulty with this procedure is that it increases the dimensionality of the training example significantly. Despite the increased size, however, the performance on the task improved dramatically.

1.2 Common Features of Successful Applications

Referring back to Chapter 1: Section 1, in which three criteria for determining the suitability of new problems to the methods presented in this thesis were described, we can see how the tasks explored in this thesis fit into the criteria:

(1) Consecutive inputs in time must have temporal coherence: In the three main tasks examined, temporal coherence has been present. In the lane-marker tracking and hand-tracking experiments, the location of the relevant features in image_t gave strong clues of where the relevant features in image_{t+1} will be. In the fault-detection experiments, a prediction of the next time-step is compared with the actual next inputs. In this task, there is the assumption that if the next wafer does not contain anomalies, the appearance of inputs_{t+1} will be predictable from the previous inputs_t .

As discussed previously in Chapter 7, and later in Chapter 8:Section 2.2, filtering is done at the pixel level. The statement “Consecutive inputs in time must have temporal coherence” should be examined in this context. The input activations are predicted, and the differences/similarities between the predicted and actual input activations are used to create the saliency map. The saliency map is used to filter the next set of inputs on an input-by-input (or pixel-by-pixel) basis.

Note that although the filtering is done with each input individually, this *does not* mean that the expectations of each input’s value at $t+1$ are created only from that input’s values in previous time steps. For example, in the lane-marker tracking task (Chapter 2), the network’s hidden units were used for prediction; in the hand-tracking task (Chapter 4), the

network's outputs and domain knowledge were used; and in the plasma-etch anomaly detection task (Chapter 5), all of the inputs were used. In the applications explored in this thesis, when all of the inputs_t were examined, the relevant inputs_{t+1} had predictable input values (in the cases in which the unexpected inputs were de-emphasized, as in lane-marker tracking and hand-tracking). Or, they had unpredictable values in the cases in which the expected inputs were de-emphasized and the unexpected ones were emphasized (plasma-etch).

(2) *Must know whether the expected should be emphasized or de-emphasized:* In all of the applications explored, a saliency map was created by comparing the expectations of the inputs with the actual inputs. Whether the expected inputs are emphasized or de-emphasized depends on the task, and dictates how the saliency map is created. In the road-following and hand-tracking experiments, the inputs were compared with the expectations; inputs which did not match were de-emphasized. In the plasma-etch task, the unexpected features were emphasized in the saliency map, since they were more informative in detecting anomalies.

(3) *There should be a method for de-emphasizing features:* As described in the previous section (Chapter 8:Section 1.1), many ways of de-emphasizing the non-salient regions were examined; for example, see the road-following and hand-tracking tasks. Even when there is no way to explicitly de-emphasize the inputs, the expected inputs and the saliency map can be used as extra inputs to the network to guide its attention. For example, see the plasma-etch task.

1.3 Methods for Interacting with a Neural Network

The hand-tracking task addressed in Chapter 4 presented a method for incorporating domain-specific knowledge to focus attention. Knowledge about the domain, in this case the physics of the real-world coupled with the sampling rate of the camera, could be incorporated into the system without requiring the system to learn this information. The constraints imposed by the domain were used to explicitly create the saliency map, instead of requiring the system to model these constraints from examples.

Because the saliency map can be explicitly modified, it can be used as a tool for interacting with other knowledge sources, including human users. These other knowledge sources can create or augment the saliency map. For example, during simulation mode, one can imagine being able to “point” at the saliency map, or at particular regions in the image, to provide suggestions about where to focus attention. This information can be incorporated on-the-fly. When such information is not available, the network can be run autonomously using the attention models developed here. In training mode, knowledge sources can be used to focus attention and aid in developing the relevant feature detectors. In training, the important features are emphasized, thereby giving the network an indication of which features to concentrate upon. Examples of this were shown in Chapter 4.

1.4 Visualizing the Contents of a Neural Network

One of the largest drawbacks of neural networks is that there are few methods of understanding what information the network has encoded or to which features the network is paying attention [Craven & Shavlik, 1995][Thrun, 1995]. This thesis has presented two methods for deriving the network’s implicit saliency map. These can be used for visualizing the contents of a neural network. The first method is to analyze what the network has encoded in the hidden layer when it is trained to solve a particular task. As described in Chapter 2, this can be accomplished by first training a network with a hidden layer to solve a task. Second, all of the weights in this network are frozen and another output layer is added (with connections only to the hidden units, as shown in Chapter 2:Figure 2-2). This output layer is used to reconstruct the inputs. However, since only the connections between the hidden layers and this new output layer are modifiable (the others are frozen), only features which are encoded in the network’s hidden units will be accurately reconstructed. In visual domains, this provides a simple and effective way to determine what the network has encoded. For example, in the lane-marker detection task, since the network only encoded information about the lane-marker position, trees or passing cars which may be present in the input image will not be encoded. Therefore, these features will not be accurately reconstructed. In summary, these reconstructed images depict

those features that the network considers relevant for performing the task. This can be used as a powerful visualization tool. Examples of this were shown in Chapter 2 and Chapter 3.

In Chapter 6, a second method to visualize the implicit focus of attention in a network was explored. In this method, a network is first trained to solve a task. Second, in a separate testing set, a small $N \times N$ block of the inputs is replaced with incorrect/noise values in every example (see Chapter 6 for more details on how to set these values). Third, these modified examples are used as inputs into the network and the error is measured. Starting with the original testing set, this process is repeated on another $N \times N$ block until all portions of the input have at some time been covered. If, when the block is replaced with noise, the error increases over using the unmodified testing set, this reflects the network's reliance on particular values being present in that portion of the input. However, if the error does not increase or is relatively small (in comparison to the increase obtained by changing other portions of the input), this indicates that the network is not paying attention to that block. This method yields easily understandable results for many vision-based tasks. Also shown in Chapter 6 is how the implicit focus of attention changes by adding noise during training.

1.5 The Role of Neural Networks for Expectation-Based Selective Attention

This study has heavily employed neural networks. However, it should be noted that many of the techniques explored in this thesis can be used with other statistical tools as well. Neural networks were used because they create a hidden layer which contains task-specific components of the inputs. However, other methods, including standard statistical regression procedures, reveal the task-specific importance of inputs. Therefore, task-specific expectations can be created by using only the inputs found to be important for the task. However, it should be noted that potentially different predictions will be made when compared to the methods explored in this thesis. This is because other statistical tools typically focus on individual inputs rather than the high level features created from multiple inputs, as may be present in the hidden units of a neural network.

It is interesting to note that, although the methods described in the previous paragraph capture a form of relevance, it is *not* dynamic. Therefore, it is difficult to use it for focusing attention in many visual processing tasks. For example, consider tracking a moving hand which appears in all regions of the input with equal probability. A static measure of input relevancy will not reveal useful information. A dynamic relevance assessment is required, as discussed in Chapter 7.

2. FUTURE WORK

There are numerous directions and application possibilities of expectation-based filtering which were not explored in this thesis. Several ideas for extending the presented work are described below.

2.1 Pixel-Based Filtering

In the current implementation, the inputs are filtered based upon their expected values. Unfortunately, this type of filtering is not suitable for all problems. For example, consider the following task. A sequence of images contain a single bar. In successive frames, it is oriented horizontally, vertically, horizontally, etc. If the bar appears in random locations in each frame, what is required to capture this with expectations, such that an anomaly would be triggered if two consecutive frames with a bar in the same orientation appeared? Individual input-based filtering is difficult in this domain since the random location of the bar will not allow the prediction of individual input activations.

An alternate implementation of this idea is to filter the *hidden* layer instead of the inputs. Rather than predicting the activations of the next inputs, the next time step's activations of the hidden layer would be predicted. By filtering at the hidden layer, processing is directed towards the feature level rather than at the pixel level. As noted in Chapter 7, however, the features in the hidden layers may not correspond to intuitive, easily understandable, objects/features in the input layer. A potential problem with this methods is that it is not known *a priori* what the best way to perform this filtering will be. Simply attenuating the activation of the hidden units in proportion to their difference from the

predicted activation may not work on all problems. On the other hand, in problems such as autonomous vehicle control with the ALVINN system [Pomerleau, 1993] where each hidden unit votes for a steering direction, this method may be very well suited. Other methods may include having the predicted hidden units as extra inputs or extra hidden units.

The above method departs from the paradigm of filtering at the pixel level. Ideally, in visual domains, filtering should be done at the *object* level. It seems more plausible that humans perform filtering at the object level rather than at the pixel level, as proposed in this thesis. It has been hypothesized that the role of attention may not only be to spotlight distinctive parts of the image, but also to segment the image into objects or parts of objects [Hurlbert & Poggio, 1986]. However, in general, this type of filtering will require sophisticated object detection procedures to analyze the scene and find the relevant objects. Unfortunately, object detection procedures are themselves fairly complicated (see for example, [Rowley, Baluja & Kanade, 1996][Sung & Poggio, 1994][Valliant *et al.*, 1994][Yang & Huang, 1994]), and are topics of ongoing research.

2.2 Parallel vs. Serial Search

The models presented in this thesis have assumed that the entire input retina is processed at the same resolution and is processed in parallel. Models which serially scan over the input retina were not used. Nonetheless, the integration of serial models may yield more biologically plausible results and increase the scenarios in which expectation-based methods can be applied. For a meaningful integration, however, many biological and psychophysical studies will need to be considered. For example, [Ullman, 1984] has suggested that the perception of spatial relations is achieved by the application of visual routines to a base representation which was formed by bottom-up processing (such as the primal or 2.5D sketch by [Marr, 1976][Marr & Nishihara, 1978]). These visual routines are composed of sequences of elemental operations. The same set of visual routines are not always applied, rather visual routines are selected to meet specific computation goals (top-down information). [Rabbit, 1984] suggests that long-term-memory (LTM) and very fast indexing into LTM plays a crucial role for deciding search paths; people very quickly

classify a novel scene into a class of previously encountered scenes for which scan paths have been developed. [Wickens, 1984] reviews ways to measure efficiency of allocating attention resources when competing demands are present.

Computational models of serial search have also been presented. For example, [Yamada & Cottrell, 1995] have explored scan paths for face recognition. [Martin *et al.*, 1993] have used saccade information for multiple character recognition. [Rimey & Brown, 1990] have used Hidden Markov Models to store and reproduce eye movements in response to remembered images. Many of the active vision paradigms described in Chapter 7, such as [Rimey & Brown, 1992], are also examples of serial search. Recently, [Rao *et al.*, 1996] have tried to model saccadic targeting during visual search.

2.3 Stability in Training

One of the problems mentioned in Chapter 2 was the potential instability in the training procedure. The feedback to the input layer can make training the system difficult. The convergence properties of this system are not known, although all of the experiments performed have quickly converged. Recently, procedures which are not based upon derivatives have been proposed as alternatives for training neural networks instead of the traditional derivative based procedures such as backpropagation. Some non-derivative based methods which have been used for setting the weights in neural networks include simulated annealing (SA) [Kirkpatrick *et al.*, 1983][Ingber & Rosen, 1992], genetic algorithms (GAs) [Holland, 1975][De Jong, 1975][Kitano, 1990], population-based incremental learning (PBIL) [Baluja & Caruana, 1995][Baluja, 1996], and evolutionary programming (EP) [Fogel, 1994][Fogel *et al.*, 1990].

These methods may be well suited for training recurrent networks since they do not require derivatives to be propagated through time; rather, they use discrete evaluations of the network. These methods work as follows: (1) weights (and possible network architecture) are randomly set, (2) the system is simulated through the training set, (3) the weight (and possibly architecture) are evaluated based upon performance on the training set, (4) based upon these evaluations (not on the error of each output – rather the

summed error of all of the outputs), new weights are selected, (5) the process is repeated from step (2). Step (4) is where each of the listed algorithms differ from each other. Step 4 can involve randomly perturbing the weights, using crossover between multiple networks, or selecting weights based on probability distributions – no derivative information is used. These methods seem well suited to this training since they do not require the propagation of errors through time and avoid the usual problems of training a dynamic system. However, a potential drawback is the large computational expense, which often prohibits the use of these techniques for many real-world tasks.

2.4 Incorporating More State Information

The systems implemented in this thesis have only used the single previous time-step to make predictions of the next inputs. However, this is not a limitation of these procedures. There are several methods of incorporating more state information. For example, in the lane-marker tracking task in Chapter 2, instead of only using the current image to predict the next image, the current actions, or multiple previous actions could also have been used. Additionally, multiple previous images could have been easily employed for making predictions.

In the hand-tracking task of Chapter 4, the hand location from several previous images could have been used to make future predictions. These predictions could have been combined with either neural networks or more traditional recursive estimation procedures such as Kalman Filters.

Lastly, standard recurrent networks, as described in Chapter 2:Section 3, can be used for maintaining state, making predictions, and estimating the task outputs. For examples, see [Jordan, 1989][Stornetta, 1988][Mozer, 1989].

2.5 Experiments with Human Expectation-Based Selective Attention

The model of attention presented here was not created to be biologically plausible. For example, it is unlikely that predictions about individual pixels in the retina are made for

the attention process. Nonetheless, an interesting topic for further study is examining the effects of expectation on selective attention in vision and other senses.

First, the effects of the *a priori* knowledge and expectations that humans bring with them to experiments could be very interesting to study. In this work, the network started as a “blank slate”. The network did not have any *a priori* knowledge of the task, the important features, or how the features would change with time. Everything was learned. This is an important point since the expectations, which are used for focusing attention, are learned simultaneously with the task. However, in humans, this may not be the case for many tasks. For example, if one studies the scan paths and learning times of humans as they are told to recognize a set of human faces, in comparison to how they are taught to recognize other objects, it is likely that a large difference will be found. For example, it has been suggested there is strong evidence that humans have an *a priori* ability to extract structure from faces – even at the early age of 37 minutes [Morton & Johnson, 1991]. The *a priori* knowledge/expectation of where important features for recognition are located in familiar object sets may lead to shorter recognition times than for unfamiliar object sets. See [Ullman, 1980] [Rabbit, 1984].

Other interesting studies may take the form of placing someone in situations for which they either have, or have been given, expectations of how the experiment will progress. The response times to anticipated and unanticipated events can be compared to subjects who were not given expectation-based clues. In this way, the role of expectations can be examined. For example, in the classic task of finding a “Green-T” in the midst of “Green-X”s and “Red-T”s [Triesman, 1988], if the location of the marker in one frame gives an indication of where the marker is likely to appear in the next frame, search times could dramatically decrease after the marker is initially found. If the experiments are to be made similar to those conducted here, it will be interesting to construct tasks in which the response given at time_t gives hints for what to focus upon in time_{t+1}. These response times can be compared to experiments in which expectations for the entire experiment are given before the experiment has started. It has been shown in a variety of experiments that setting up expectations of the subjects about what they will perceive has a large impact on what is actually perceived [Kagan & Havemann, 1976].

With respect to the heavily investigated question of if/when/where filtering takes place, the model presented here works as follows. Initially, all of the inputs are processed. From this point, expectations of what the next inputs will be are formed. These expectations control the filtering process in the next time-step. The next inputs are filtered based upon the expected inputs and the actual input (at a low-level) before the inputs are processed. The remaining inputs are processed (at a higher level), and the next expectations are formed. An interesting study to perform will be to examine the sensitivity of perception in the regions in which the most activity/saliency for the task is expected by a subject, in comparison to the sensitivity in the regions in which no activity/saliency was expected¹. One experiment to test this would be to have a subject perform a task with temporal structure – such that the place to focus at time_{t+1} depends on the location of the salient features at time_t. Distraction features/markers are placed elsewhere in the scene. At random times, the screen is blanked and the subject is asked to recall all that they saw in the scene. Response accuracy may be different for the markers within the salient regions and those outside of the salient regions. In particular, the responses may be better for markers where the next marker was expected to appear, rather than where it was in the previous frame (depending on the time scales at which the scenes change). Experiments which use visual cues to draw attention are described in [Posner & Cohen, 1994] and are reviewed in [Kellogg, 1995].

This experiment can be studied along at least three axes. First, the accuracy of responses can be measured as the proximity of the distraction markers to the salient region is varied. If distraction markers are put closer to the salient marker, does this force a tighter focus of attention? This will reveal information about the size of the attentional spotlight, see for example [Murphy & Eriksen, 1987].

Second, response accuracy can be examined as the accuracy of the expectations degrade. For example, the location of the salient feature in time_t may indicate exactly where the next salient region will be, a small region where the next salient marker may be, or just which half of the scene in which the next salient feature will appear. This will also reveal information about the size of the attentional spotlight.

1. A distinction should be made between this study and studying foveated vs. peripheral vision, since focus of attention does not necessarily entail eye movements.

Third, another interesting test would be to examine the effects of using spatially continuous vs. discontinuous (but always predictable) regions of saliency – much like the discontinuous cross detection tasks in Chapter 3. In the model presented here, unlike traditional “spotlight theories”, the spotlight in these experiments can be in multiple regions concurrently, and it does not necessarily move continuously over time.

2.6 Future Application Areas

Four applications were explored in this thesis. The first three used the expectation-based selective attention methods: lane-tracking for autonomous monitoring and control of a land vehicle, hand-tracking in cluttered scenes, and the detection of anomalies in the plasma-etch step of semiconductor wafer fabrication. The fourth application, the detection of faces in arbitrary scenes, used the techniques created in this thesis for analyzing the trained neural-networks. These analysis techniques are general, and many vision-based tasks which employ neural networks can use them for intuitive explanations of the features to which the network is attending.

There are numerous other applications which can be addressed using the expectation-based filtering. For example, speech recognition is a domain which has an important temporal component. Expectations can be used to de-emphasize noise in the inputs. It is easy to imagine connecting another output layer to the hidden layer of the time-delay neural networks (TDNN) used in [Waibel *et. al.*, 1989] to predict the next inputs, as in Chapters 2-3. However, in contrast to the domains explored in this thesis, it is likely that speech recognition may require many previous frames to accurately predict the next frame.

A second domain for future exploration is automatic obstacle detection for autonomous road following. Chapters 2 and 5 described a method for obstacle detection based upon the unexpected portions of the input. The predictions of the future inputs are compared with the actual inputs. The points of difference are designated salient. This was not implemented in the vision-based system described in Chapter 2 since alternate sensors (not based on vision) may be more useful than cameras for this task. Nonetheless, since these sensors give feedback which is temporally related in the same way as vision-based

frames, expectations can also be used to predict the next inputs, whether the next inputs are laser-range-data, sonar-data, or intensity data. As shown in Chapter 5, the methods explored in this thesis are not limited to visual domains.

A third domain is vision-based gesture-recognition; Chapter 4 presented a system which was a step in this direction. The presented system can easily be extended. Instead of using expectation to search around the area in which the hand was previously found, information about movements associated with specific gestures can guide the expectations. In domains in which a limited number of gestures need to be recognized, watching a gesture in progress can aid in narrowing the next search region.

As more complex domains are addressed, the need to use all of the available information will increase. Expectation-Based Selective Attention is a method which allows learning procedures to exploit the temporal coherence and predictability of many real-world tasks. In the light of limited training data and resources, this extra source of information can prevent learning algorithms from focusing on irrelevant or misleading features.

REFERENCES & BIBLIOGRAPHY

- Alumualim, H. & Dietterich, T.G., (1991) "Learning with Many Irrelevant Features", in: *Proceedings of the Ninth International Conference on Artificial Intelligence*. San Jose, CA, AAAI Press, 547-552.
- Ahmad, S. (1991), *VISIT: An Efficient Computation Model of Human Attention*. Ph.D. Thesis. University of Illinois at Urbana Champaign.
- Ahmad, S. & Omohundro, S. (1990) "Efficient Visual Search: A Connectionist Solution". In *Proceedings of the 13th Annual Conference of the Cognitive Society, Chicago*, 1991. Also ICSI-TR-91-040.
- Allport, A. (1989) "Visual Attention", in: Posner, M., ed., *Foundations of Cognitive Science*, MIT Press, Cambridge, Ma., 631-683.
- Aloimonos, J., Weiss, I, Bandyopadhyay, A. (1988) "Active Vision", *International Journal of Computer Vision*. 1:4 pp. 333-356.
- Baldi, P. & Hornik, K. (1989), "Neural Networks and Principal Components Analysis: Learning from Examples without Local Minima", *Neural Networks* 2 (1989) 53-58.
- Ballard, D. & Brown, C.M. (1982) *Computer Vision*, Prentice-Hall, Inc. New Jersey.
- Ballard, D. & Ozcandarli, A. (1988) "Eye Fixation and Early Vision: Kinetic Depth", in (eds.) Bajcsy, R. & Ullman, S., *The Proceedings of the International Conference on Computer Vision-2*. IEEE Computer Society, Washington, D.C.
- Baluja, S. (1996), "Evolution of an Artificial Neural Network Based Autonomous Vehicle Controller", in *IEEE Transactions on Systems, Man and Cybernetics, part B: Cybernetics*. June 1996, Vol. 26, No. 3, pp. 450-463.
- Baluja, S. (1994) "Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning" CMU-CS-94-163. Available via anonymous ftp at: reports.adm.cs.cmu.edu also <http://www.cs.cmu.edu/~baluja>.

- Baluja, S. & Caruana, R. (1995) "Removing the Genetics from the Standard Genetic Algorithm". In (eds) Frieditis, A., Russel, S. *The International Conference on Machine Learning 1995 (ML-95)*: Lake Tahoe, California. 1995. Morgan Kaufmann Publishers. San Mateo, CA. pp. 38-46. Report is available on-line from: <http://www.cs.cmu.edu/~baluja>.
- Baluja, S. & Maxion, R. (1996) "Artificial Neural Network Based Detection and Classification of Plasma-Etch Anomalies". To appear in: *The Journal of Intelligent Systems*.
- Baluja, S. & Pomerleau, D.A., (1995), "Using a Saliency Map for Active Spatial Selective Attention: Implementation and Initial Results", in: (eds.) Tesauro, G., Touretzky, D.S., Leen, T.K., *Advances in Neural Information Processing Systems 7*. MIT Press, Cambridge MA, 451-458.
- Baluja, S. & Pomerleau, D.A., (1995b), "Using the Representation in a Neural Network's Hidden Layer for Task-Specific Focus of Attention", in: *Proceedings International Joint Conference on Artificial Intelligence-95. IJCAI*, San Mateo, CA., 133-139. Report is available on-line from: <http://www.cs.cmu.edu/~baluja>.
- Blake, A. & Alan, Y. (1992) (editors) *Active Vision*. MIT Press, Cambridge, MA.
- Bobick, A.F. & Wilson, A.D. (1995) "A State-based Technique for the Summarization and Recognition of Gesture" in *Proceedings of the Fifth International Conference on Computer Vision*. IEEE Computer Society Press. Washington.. pg. 382-388.
- Bolt, G., 1991, "Investigating Fault Tolerance in Artificial Neural Networks", University of York - Department of Computer Science Technical Report (YCS-154).
- Bolt, G., Austin, J., Morgan, G., 1992, "Fault Tolerant Multi-Layer Perceptron Networks", University of York - Department of Computer Science Technical Report (YCS-180).
- Broadbent, D.E., (1971) *Decision and Stress*, Academic Press, London.
- Broadbent, D.E., (1982) "Task Combination and Selective Intake of Information", in: *Acta Psychologica* 50 253-290.
- Caruana, R. (1993), "Multitask Learning: A Knowledge-Based Source of Inductive Bias", in (ed.) Utgoff, P. *Machine Learning: Proceedings of the Tenth International Conference*. San Mateo, CA: Morgan Kaufmann. pp. 41-48.
- Caruana, R., Baluja, S., Mitchell, T. (1996), "Using the Future to 'Sort-Out' The Present: Rankprop and Multitask Learning for Medical Risk Evaluation," in (eds.) Touretzky, D.S., Mozer, M.C., & Hasselmo, M.E. *Advances in Neural Information Processing Systems 8*. MIT Press. 959-965.
- Caruana, R. & Freitag, D. (1994), "Greedy Attribute Selection", in: *Proceedings of the Eleventh Interna-*

- tional Conference on Machine Learning*, Morgan Kaufmann, San Mateo, CA. 28-36.
- Clark, J. & Ferrier, N (1992), Attentive Visual Servoing, in: *Active Vision*. A. Blake & A.Yuille, eds., (MIT Press, Cambridge, MA) 137-154.
- Clay R.C. & Sequin C.H., 1990, "Fault Tolerance in Artificial Neural Networks", *1990 IJCNN: International Joint Conference on Neural Networks*. IEEE. I-703,708.
- Clay R.C. & Sequin C.H., 1992, "Fault Tolerance Training Improves Generalization and Robustness", *1992 IJCNN: International Joint Conference on Neural Networks*. IEEE. I-769-774.
- Cottrell, G.W., (1990) "Extracting Features from Faces using Compression Networks: Face, Identity, Emotion and Gender Recognition Using Holons", in: *Connectionist Models: Proceedings of the 1990 Summer School*, Morgan Kaufmann, San Mateo, CA., 328-337.
- Cottrell, G.W. & Munro, P. (1988) "Principal Component Analysis of Images via back-propagation." *Proc. Soc. of Photo-Optical Instr. Eng.*, Cambridge, MA.
- Craven M.W., & Shavlik, J.W., (1996) "Extracting Tree-Structured Representations of Trained Networks" in (eds.) Touretzky, D.S., Mozer, M.C., & Hasselmo, M.E. *Advances in Neural Information Processing Systems* 8. MIT Press. 24-30.
- Cui, Y. & Weng, J.J. "Hand Segmentation Using Learning-Based Prediction and Verification for Hand Sign Recognition." in *Computer Vision and Pattern Recognition*, 1996. IEEE Computer Society, CA. 88-93.
- Darrell, T., Moghaddam, B., Pentland, A.P, "Active Face Tracking and Pose Estimation in an Interactive Room", in *Computer Vision and Pattern Recognition*, 1996. IEEE Computer Society, CA. 67-72
- De Jong, K (1975) *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, Doctoral Dissertation, Dept. Computer and Communication Science. University of Michigan, Ann Arbor.
- Deutsch, J.A., & Deutsch, D. (1963) "Attention: Some Theoretical Considerations", *Psychological Review*, Vol.70, No.1, 80-90.
- Dickmanns, E. "Expectation-based Dynamic Scene Understand" (1992), in (eds.) Blake & Yuille, *Active Vision*, MIT Press, Cambridge Massachusetts, pp. 303-334.
- Elman, J.L. (1990) "Finding Structure in Time". *Cognitive Science* 14, 179-211.
- Fogel, D.B., (1994) "An Introduction to Simulated Evolutionary Optimization", *IEEE Transactions of Neural Networks*, Vol. 5, no.1, pp 3-13, 1994.

- Fogel, D.B., Fogel, L.J., Porto, W. (1990) "Evolving Neural Networks", *Biological Cybernetics*, vol. 63, pp 487-493, 1990.
- Fogelman-Soulie, F. (1995) "Applications of Neural Networks" in Arbib, M. (ed). *The Handbook of Brain Theory and Neural Networks*. MIT Press. Cambridge, Mass. pp. 94-98.
- Fukushima, K. "Neural Network Model for Selective Attention in Visual Pattern Recognition and Associative Recall". *Applied Optics*. Vol 26:23. pp 4985-4992.
- Girosi, F., Jones, M., & Poggio, T., (1995) "Regularization Theory and Neural Network Architectures", *Neural Computation*, 7:219-269.
- Goddard, N.H. (1992) *The Perception of Articulated Motion: Recognizing Moving Light Displays*. Ph.D. Thesis, University of Rochester.
- Hampshire, J.B. & Waibel, A.H. (1989) "A Novel Objective Function for Improved Phoneme Recognition using Time-Delay Neural Networks". CMU-CS-89-118. School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.
- Hassibi, B. & Stork, D.G. (1993), "Second Order Derivatives for Network Pruning: Optimal Brain Surgeon" in: *Advances in Neural Information Processing Systems 5*, Morgan Kaufmann Publishers, San Mateo, CA., 164-171.
- Hassibi, B., Stork, D.G., Wolff, G. & Watanabe, T., (1994) "Optimal Brain Surgeon: Extensions and Performance Comparisons", in: *Advances in Neural Information Processing Systems 6*, Morgan Kaufmann Publishers, San Mateo, CA., 263-270.
- Hertz, J., Krogh, A., & Palmer, R., (1989) *Introduction to the Theory of Neural Computation*, Addison-Wesley.
- Hinton, G.E. (1987) "Connectionist Learning Procedures". Technical Report CMU-CS-87-115 (Version 2). Carnegie Mellon University, Computer Science Department, Pittsburgh, PA. Also appeared in the *Artificial Intelligence Journal*.
- Hoffman, J. (1986) "Spatial Attention in Vision: Evidence for Early Selection", in *Psychological Research* 48: 221-229.
- Holland, J. H. (1975) *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: The University of Michigan Press.
- Howarth, R. & Buxton, H. (1993) "Selective Attention in Dynamic Vision". in: *The Proceedings of the International Joint Conference on Artificial Intelligence-13*. 1579-1584.

- Hurlbert, A. & Poggio, T., (1985) "Spotlight on Attention", MIT AI Laboratory Memo AI- 817. Massachusetts Institute of Technology, Cambridge, MA. (1985).
- Hurlbert, A. & Poggio, T. (1986) "Do Computers Need Attention?", *Nature*, 321:651-652.
- Ingber, L. & Rosen, B., (1992) "Genetic Algorithms and Very Fast Simulated Re-annealing: A Comparison", *Mathematical Computer Modelling*, V16, N11, 87-100.
- Japkowicz, N., Myers C., & Gluck M., (1995), "A Novelty Detection Approach to Classification", in Mellish, C. (ed.) *The International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, Canada. IJCAI & Morgan Kaufmann. San Mateo, CA. pp 518-523.
- Jacobs, R.A., Jordan, M.I, Barto, A.G. (1990), "Task Decomposition through Competition in a Modular Connectionist Architecture: The what and where vision tasks", COINS Technical Report 90-27. University of Massachusetts, Amherst.
- John, G.H., Kohavi, R., Pfleger, K. (1994), "Irrelevant Features and the Subset Selection Problem", in: *Proceedings of the Eleventh International Conference on Machine Learning*, Morgan Kaufmann, San Mateo, CA. 121-129.
- Jordan, M.I, (1989) "Serial Order: A Parallel, Distributed Processing Approach", in: J.L. Elman and D.E. Rumelhart, eds., *Advances in Connectionist Theory: Speech*, Hillsdale: Erlbaum.
- Judd, S. & Munro, P. W., 1993, "Nets with Unreliable Hidden Nodes Learn Error Correcting Codes" *Advances in Neural Information Processing Systems 5*. Hanson, S.J., Cowan, J.D., Giles, L.C., (eds) Morgan Kaufmann Publishers. CA. pp. 89-96.
- Kaelbling, L, Littman, M. & Moore, A. (1996), Reinforcement Learning: A Survey. to appear in the *Journal of Artificial Intelligence Research (JAIR)*- 1996.
- Kagan, J. & Havemann, E. (1976) *Psychology: An Introduction (Third Edition)*. Harcourt Brace Jovanovich, Inc. New York.
- Kellogg, R.T. (1995), *Cognitive Psychology*, Sage Publishing, Thousand Oaks, California.
- Kinchla, R.A. (1992), "Attention", *Annual Review of Psychology*, 43 711-742.
- Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., "Optimization by Simulated Annealing", *Science* 220 (4598), 671-680, 1993.
- Kitano, H. (1990) "Empirical Studies on the Utility of Genetic Algorithms for the Training and Designing of Neural Networks," Tech Report CMU-CMT-90-120, Center for Machine Translation, Carnegie Mellon University, 1990.

- Koch, C. & Ullman, S. (1985) "Shifts in Selective Visual Attention: Towards the Underlying Neural Circuitry", in: *Human Neurobiology* 4 (1985) 219-227.
- Kohavi, R and John, G.R., (1995) "Automatic Parameter Selection by Minimizing Estimated Error", in: *Proceedings of the Twelfth International Conference on Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA., 303-311.
- Kramer, M. (1991) "Nonlinear Principal Component Analysis Using Autoassociative Neural Networks", in: *AIChE Journal* 37:2, 233-242.
- Krotkov, E. (1990) "Active Perception for Legged Locomotion: Every Step is an Experiment". Appeared in: *IEEE International Symposium on Intelligence Control*. Philadelphia, PA. Sept. 1990.
- Laars, P., Heskes, T., Gielen, S. (1996), "A Neural Model of Visual Attention", in: *Neural Networks: Artificial Intelligence and Industrial Applications*, B. Kappen. and S. Gielen, eds., To appear.
- Langley, P., Sage, S., (1994) "Oblivious Decision Trees and Abstract Cases", in: *Working Notes of the AAAI-94 Workshop on Case-Based Reasoning*. AAAI Press, Seattle WA, 113-117.
- LeCun, Y., Boser, B., Denker, J.S., Henderson, D. Howard, R.E., Hubbard W., and Jackel, L.D. (1990) "Handwritten digit recognition with a back-propagation network", in (ed.) Touretzky, D., *Advances in Neural Information Processing Systems* 2. San Mateo, CA: Morgan Kaufmann. pp. 396-404.
- LeCun, Y., Boser, B., Denker, J.S., Henderson, D. Howard, R.E., Hubbard W., and Jackel, L.D. (1989) "Backpropagation Applied to Handwritten Zip Code Recognition", *Neural Computation* 1, 541-551. MIT, 1989.
- Marr, D. (1982) *Vision*. W.H. Freeman and Co. New York, 1982.
- Marr, D. (1976) "Early Processing of Visual Information", *Phil. Trans. Roy Soc. and B*, 275. 483-524.
- Marr, D. & Nishihara, H.K. (1978) "Representation and Recognition of the Spatial Organization of Three-Dimensional Shapes", *Proc. Royal Society of London B*, 200, 269-294.
- Martin, G., Rashid, M., Chapman, D. Pittman, J., (1993) "Learning to See Where and What: Training a Net to Make Saccades and Recognize Handwritten Characters", in (eds) Hanson, S.J., Cowan, J.D., Giles, C.L., *Advances in Neural Information Processing Systems* 5. San Mateo, CA: Morgan Kaufmann. pp. 441-447.
- Maxion, Roy *et al.* (1996) "Real World Detection and Diagnosis of Plasma-Etch Anomalies". *In Progress*.
- Maxion, Roy A. (1990) "Toward Diagnosis as an Emergent Behavior in a Network Ecosystem." *Physica D*,

- Vol. 42, Nos. 1-3, pages 66-84. June 1990.
- Maxion, Roy and Olszewski, Robert. (1993) "Detection and Discrimination of Injected Network Faults." *23rd Annual International Symposium on Fault-Tolerant Computing*, pages 198-207. Toulouse, France; June 22-24. IEEE Computer Society Press.
- Maybeck, P.S. (1990) "The Kalman Filter: An Introduction to Concepts". in (eds.) Cox, I.J., Wilfong, G.T., *Autonomous Robot Vehicles*. Springer-Verlag, New York. pp. 194-204.
- Mitchell, T.M, O'Sullivan, J., Thrun, S. (1994) "Explanation-Based Learning for Mobile Robot Perception". *Machine Learning Conference and Computational Learning Theory Workshop on Robot Control*. Rutgers, The State University of New Jersey, New Brunswick, July 9, 1994.
- Mitchell, T. & Thrun, S., (1993) "Explanation-based neural network learning for robot control" in (eds) Hanson, S.J., Cowan, J.D., Giles, C.L., *Advances in Neural Information Processing Systems 5*. San Mateo, CA: Morgan Kaufmann. pp. 287-294.
- Miikkulainen, R., Dyer, M.G., 1988 "Encoding Input/Output Representations in Connectionist Cognitive Systems" *Proc.1988 Connectionist Models Summer School*. Morgan Kaufmann Publishers.
- Mozer, M.C. (1989) "A Focused Back-Propagation Algorithm for Temporal Pattern Recognition". *Complex Systems* 3, 349-381.
- Mozer, M.C. (1988) "A Connectionist Model of Selective Attention in Visual Perception", Technical Report CRG-TR-99-4, University of Toronto, Toronto, Canada.
- Mozer, M.C. & Smolensky, P. (1989), "Skeletonization: A Technique for Trimming the Fat from a Network Via Relevance Assessment", in: *Advances in Neural Information Processing Systems 1*, Morgan Kaufmann Publishers, San Mateo, CA., (1989) 107-115.
- Morton, J. & Johnson, M. (1991) "The Perception of Facial Structure in Infancy", in (eds) Lockhead, G.R., Pomerantz, J.R., *The Perception of Structure: Essays in Honor of Wendell R. Garner*, American Psychological Association, Washington D.C.317-325.
- Murphy, T.D. & Eriksen, C.W. (1987) "Temporal Changes in the Distribution of Attention in the Visual Field in Response to Precues", *Perception & Psychophysics*, 42, 576-586.
- Murray, A.F. & Edwards, P.J., 1993, "Synaptic Weight Noise During MLP Learning Enhances Fault-Tolerance, Generalization and Learning Trajectory", *Advances in Neural Information Processing Systems 5*. Hanson, S.J., Cowan, J.D., Giles, L.C., (eds) Morgan Kaufmann Publishers. CA. pp 491-498.
- Nayar, S.K. & Poggio, T. (1996) Editors. *Early Visual Learning*. Oxford University Press, New York.

- Nowlan, S. & Platt, J. C. (1995) "A Convolutional Neural Network Hand Tracker", in: (eds.) Tesauro, G., Touretzky, D.S., Leen, T.K., *Advances in Neural Information Processing Systems 7*. MIT Press, Cambridge MA, 901-908.
- Olshausen, N., Anderson, C., Essen, D.V. (1992), "A Neural Model of Visual Attention and Invariant Pattern Recognition", Technical Report, CNS-Memo 18, California Institute of Technology.
- Olshausen, N. & Koch, C. (1996) "Selective Visual Attention", in (ed.) M. Arbib, *The Handbook of Brain Theory and Neural Networks*. MIT Press, MA. 837-840.
- Pashler, H. & Badgio, P.C. (1985) "Visual Attention and Stimulus Identification", in: *Journal of Experimental Psychology: Human Perception and Performance* 11: 2 (1985)105-121.
- Petsche, T., Marcantonio, A., Darken, C., Hanson, S.J., Kuhn, G.M., Santoso, I., (1996) "A Neural Network Autoassociator for Induction Motor Failure Prediction", in (eds.) Touretzky, D.S., Mozer, M.C., & Hasselmo, M.E. *Advances in Neural Information Processing Systems 8*. MIT Press. 924-930.
- Poggio, T. & Beymer, D. (1996) "Regularization Networks for Visual Learning", in (eds) Nayar, S.K. & Poggio, T., *Early Visual Learning*. Oxford University Press, New York.
- Pomerleau, D.A., (1995) "RALPH: Rapidly Adapting Lateral Position Handler", in: *1995 IEEE Symposium on Intelligent vehicle*, September 25-26, 1995, Detroit, Michigan.
- Pomerleau, D.A. (1994), "Reliability Estimation for Neural Network Based Autonomous Driving", in: *Robotics and Autonomous Systems* 12, 113-119.
- Pomerleau, D.A. (1993), *Neural Network Perception for Mobile Robot Guidance*, Kluwer Academic Publishing, Boston, Massachusetts.
- Pomerleau, D.A., (1991) "Efficient Training of Artificial Neural Networks for Autonomous Navigation", in: *Neural Computation* 3:1, 88-97.
- Pope, R.H. (1986) "Human Performance: What Improvement from Human Reliability Assessment" in H.J. Wingender (ed.) *Reliability Data Collection and Use in Risk and Availability: Proceedings of the 5th EureData Conference*. Springer-Verlag, Berlin, 455-465.
- Posner, M.I, Snyder, C.R.R., Davidson, B.J., (1980) "Attention and the Detection of Signals", in *Journal of Experimental Psychology: General*. Vol. 109, No. 2, 160-174.
- Posner, M. I. & Cohen, Y. (1984) "Components of Visual Orienting", in (eds.) H. Bouma & D.G. Bouwhuis, *Attention and Performance X: Control of Language Processes*, Hillsdale, NJ, Lawrence-Erlbaum, 531-554.

- Quinlan, J.R. (1993), *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.
- Rabbit, P. (1984) "The Control of Attention in Visual Search", in (ed) Parasuraman, R. & Davies, D.R., *Varieties of Attention*. 273-291. Academic Press, Inc, Orlando.
- Rao, R.P.N., Zelinsky, G.J., Hayhoe, M.M. Ballard, D.H., (1996) "Modeling Saccadic Targeting in Visual Search", in (eds.) Touretzky, D.S., Mozer, M.C., & Hasselmo, M.E. *Advances in Neural Information Processing Systems 8*. MIT Press. 830-836.
- Reed, R. & Marks, R.J. (1995) "Neurosmithing: Improving Neural Network Learning", in (ed.) M. Arbib, *The Handbook of Brain Theory and Neural Networks*. MIT Press, MA. 639-643.
- Rehg, J. & Kanade, T. (1995) "Model-Based Tracking of Self-Occluding Articulated Objects", in *Proceedings of the Fifth International Conference on Computer Vision*. IEEE Computer Society Press. Washington.. pg.612-617.
- Rimey, R. & Brown, C. (1990) "Selective Attention as Sequential Behavior: Modeling Eye Movements with an Augmented Hidden Markov Model", Technical Report. University of Rochester.
- Rimey, R. & Brown, C. (1992a) "Task-oriented Vision with Multiple Bayes Nets" (1992), in (eds.) Blake & Yuille, *Active Vision*, MIT Press, Cambridge Massachusetts, pp. 217-236.
- Rimey, R. & Brown, C. (1992b) "Where to Look Next Using a Bayes Net: Incorporating Geometric Relations", in (ed) Sandini, G. *European Conference on Computer Vision*, Berlin. 542-550.
- Rowley, H., Baluja, S., Kanade, T. (1996), "Human Face Detection in Visual Scenes," in (eds.) Touretzky, D.S., Mozer, M.C., & Hasselmo, M.E. *Advances in Neural Information Processing Systems 8*. MIT Press. 875-881. See also "Neural Network Based Face Detection" in *Computer Vision and Pattern Recognition*, 1996. IEEE Computer Society, CA. 203-208.
- Rowley, H., Baluja, S., Kanade, T. (1995), "Human Face Detection in Visual Scenes", Technical Report. CMU-CS-95-158R. School of Computer Science, Carnegie Mellon University. November, 1995. Report is available on-line from: <http://www.cs.cmu.edu/~baluja>.
- Rumelhart, D.E., Hinton, G.E., and Williamson, R.J. (1986). "Learning Internal Representations by error propagation." In *Parallel Distributed Processing*. Bradford Books, MA.
- Russ, M. 1985. *Plasma Etching in Semiconductor Fabrication*. Elsevier: Amsterdam.
- Schmidhuber, J. & Huber, R. (1991) "Learning to generate artificial fovea trajectories for target detection." *International Journal of Neural Systems*, 2(1 & 2):135-141.
- Sietsma, J. & Dow, J.F., 1991 "Creating Artificial Neural Networks that Generalize", *Neural Networks* 4,.

67-79.

- Simard, P., Victorri, B., LeCun, Y., & Denker, J. (1992) "Tangent Prop - A Formalism for Specifying Selected Invariances in an Adaptive Network". In (eds.) Moody, J.E., Hanson, S.J., Lippman, R.P. *Advances in Neural Information Processing Systems 4*. San Mateo, CA: Morgan Kaufmann. pp. 895-903.
- Staddon, J.E.R., King, M., Lohead, G.R. (1980) "On Sequential Effects in Absolute Judgment Experiments", *Journal of Experimental Psychology: Human Perception and Performance*, vol. 6, No.2 290-301.
- Stornetta, W.S., Hogg, T., & Huberman, B.A. (1988), "A Dynamical Approach to Temporal Pattern Processing", in: *Neural Information Processing Systems*, NY 750-759.
- Stroop, J.R. (1935) "Studies of Interference in Serial Verbal Reactions", *Psychological Monographs*, 50, 38-48.
- Sung, K.K. & Poggio, T. (1994) "Example-Based Learning for View-Based Human Face Detection" A.I. memo 1521, CBCL Paper 112. Massachusetts Institute of Technology, December, 1994.
- Swain, M. (1994) (editor) *International Journal of Computer Vision* - Special Issue on Active Vision II. Volume 12, Numbers 2/3, April, 1994.
- Swain, M. (1993) (editor) *International Journal of Computer Vision* - Special Issue on Active Vision I. Volume 11, Number 2, October, 1993.
- Swain, M.J. & Stricker, M.A. (1993) "Promising Directions in Active Vision", in *International Journal of Computer Vision - Special Issue on Active Vision I*. Volume 11, Number 2, October, 1993.
- Tebelskis, J. (1995) *Speech Recognition Using Neural Networks*. Ph.D. Thesis. CMU-CS-95-142. School of Computer Science, Carnegie Mellon University, Pittsburgh, PA. Available via anonymous ftp at reports.adm.cs.cmu.edu. 80-84.
- Thorpe, C., (1991) "Outdoor Visual Navigation for Autonomous Robots", in: *Robotics and Autonomous Systems* 7, 85-98.
- Thrun, S. (1995) "Extracting Rules from Artificial Neural Networks with Distributed Representations", in: (eds.) Tesauro, G., Touretzky, D.S., Leen, T.K., *Advances in Neural Information Processing Systems 7*. MIT Press, Cambridge MA, 505-512.
- Trick, L.M. & Pylyshyn, Z.W., (1991) "Preattentive Indexing and Visual Counting: FINSTs and the Enumeration of Concentric Items", Technical Report Cogmem #58, University of Western Ontario.

- Triesman, A. & Gelade, G. (1980) "A Feature Integration Theory of Attention", in *Cognitive Psychology*: 12, (1980) 97-136.
- Triesman, A. (1988) "Features and Objects: The Fourteenth Bartlett Memorial Lecture", *Q.J. Exp. Psychol [A]*, 40:201-237.
- Tsotsos, J.K. (1992) "On the Relative Complexity of Active vs. Passive Visual Search", *International Journal of Computer Vision*, Volume 7, No. 2. 127-142.
- Ullman, S., (1980) "Visual Routines", in: (ed.) Pinker, S., *Visual Cognition*. MIT Press, Cambridge, Massachusetts, 97-160.
- Umiltà, C., (1988) "Orienting of Attention", in: (eds.) Boller, F. & Grafman, J., *Handbook of Neuropsychology* V1. Elsevier, Amsterdam. 175-193.
- Ungar, L. (1995) "Forecasting" in Arbib, M. (ed). *The Handbook of Brain Theory and Neural Networks*. MIT Press. Cambridge, Mass. pp. 399-403.
- Valliant, R., Morocq, C., Le Cun, Y. (1994) "Original Approach for the Localization of Objects in Images", *IEE Proceedings on Vision, Image, and Signal Processing*, 141(4) August. 1994.
- Van der Heijden, A.H.C. (1992), *Selective Attention in Vision*. Routledge, London.
- Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., Lang, K. (1989) "Phoneme Recognition using Time-Delay Neural Networks". In A. Waibel & K.F.Lee (eds), *Readings in Speech Recognition*, pp. 393-404. Morgan Kaufmann Publishers. CA. Also appears in *IEEE Transactions on Acoustics, Speech and Signal Processing* 37, 328-339.
- Wickens, C.D. (1984), "Processing Resources in Attention", in (ed) Parasuraman, R. & Davies, D.R., *Varieties of Attention*. 63-101. Academic Press, Inc, Orlando.
- Widrow, B. & Lehr, M.A. (1995) "Noise Canceling and Channel Equalization" (ed.) M. Arbib, *The Handbook of Brain Theory and Neural Networks*. MIT Press, MA. pp. 648-650.
- Yamada, K & Cottrell, G.W., "A Model of Scan Paths Applied to Face Recognition", in *Proc. 17th Annual Conf. of the Cognitive Science Society*, 1995.
- Yang, G. & Huang, T.S. (1994), "Human Face Detection in a Complex Background", *Pattern Recognition*, 27(1):53-63.