UNIVERSITY OF CALIFORNIA,

IRVINE

Modular Neural Network Architecture for Detection of Operational Problems on Urban Arterials

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTORAL OF PHILOSOPHY

in Civil Engineering

by

Sarosh Islam Khan

Dissertation Committee:
Professor Stephen Ritchie

1996

The dissertation of Sarosh Islam Khan
is approved and is acceptable in quality
and form for publication on microfilm:

_____
_____
_____
Committee Chair

University of California, Irvine
1996

MODULAR NEURAL NETWORK ARCHITECTURE FOR DETECTION OF OPERATIONAL PROBLEMS ON URBAN ARTERIALS

LIST OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

1.1 Problem Definition

In recent years, transportation research has revealed that problems of widespread congestion cannot be solved by building more roads or by expanding existing infrastructure. A significant part of the solution lies in better management of traffic. One of the principal thrusts of the new national program on Intelligent Transportation Systems (ITS) is Advanced Transportation Management Systems (ATMS).

To facilitate better management, recent research has focused on continuous monitoring of traffic to ascertain the 'normal' level of congestion and to provide an understanding of how it forms and spreads. Techniques for rapidly detecting incidents have become a vital link in the management of traffic. As pointed out by Ritchie (1990), a major concern in ATMS is providing decision support to effectively detect, verify and develop response strategies for incidents that disrupt the flow of traffic. A key element of providing such support is automating the process of detecting operational problems on large area networks. Successful detection of operational problems in their early stages is vital for formulating response strategies such as modifying surface street signal timing plans and activating or updating traveler information systems, including changeable message signs, in-vehicle navigation systems and highway advisory radio, amongst others. It is also needed as a basis for alerting police, emergency vehicles and tow services. Reliable

surface street incident detection is necessary for the development of integrated freeway-arterial control systems, and will permit improved coordination of freeway ramp meters and surface street signal timing. Therefore, developing a capability for automating incident detection on arterial streets will aid in accomplishing a true integration of freeway and arterial networks.

From the early 1970's, research has been conducted to develop incident detection algorithms for freeways to aid traffic engineers. Only recently, since the mid 1980's, has any research focused on similar efforts for surface streets. As pointed out by Han and May (1988), little work has been done on the arterial side because of characteristic differences between freeways and arterials. Therefore a significant challenge lies in formulating an incident detection methodology for arterial streets.

Differences between freeways and surface streets that impact incident detection include the following:

- *multiple access*: freeways have directed access points through entry and exit ramps, but surface streets have multiple access points through left, right and through movements of traffic, giving drivers multiple choices

- *geometric constraints*: surface streets have geometric constraints such as channelization for separation of traffic movements, and conflicting movements; freeways have fewer of these features

- *control measures*: entry ramps on freeways control the rate of traffic entering the main line flow, whereas intersection control on surface street networks controls phase sequencing through either fixed time control or traffic actuated control with variable splits, depending on the local demand within the constraints of cycle lengths, and minimum and maximum phase lengths.

- *operating conditions*: surface streets usually operate at lower speeds compared to freeways, which allows vehicles on surface streets to change lanes more readily, thereby making the problem of distinguishing between incident and non-incident patterns more difficult as vehicles are able to maneuver around incident locations more easily.

- *detector configuration*: freeways usually have more uniform spacing of detector stations (e.g. half or one-third of a mile), but for surface streets the surveillance detector location varies based on the length of the links

Because of the characteristic differences between surface streets and freeways, the effects, types and nature of 'incidents' differ. Very limited work has been done in developing an algorithm to detect incidents on signalized surface street networks. Of the few attempts (Bell and Thancanamootoo, 1988; Han and May, 1989; Chen and Chang,, 1993), none have been extensively tested or implemented for a city's street network.

Therefore, there is a need for the development of an incident detection system for signalized street networks that will automate the process for a traffic management center.

This research proposes to develop such a new approach to detecting incidents or traffic operational problems on surface streets.

1.2 Research Approach

The objective of developing an algorithm to automate the process of detecting operational problems or traffic management problems is to provide traffic management centers, overseeing the operations and control of street networks, with a decision tool. This tool, embedded in a traffic management system, would be part of a four step incident management system - incident detection, incident confirmation, incident response and recovery monitoring (Ritchie, 1990). In the case of surface street networks, any operational problem that requires the attention of an operator in a traffic management center, and results in an operator formulating a response, mey be defined as an incident. Therefore, the role of a detection algorithm will be to detect the following types of incidents that are relevant to the operation and control of surface street networks:

- Reduced capacity:

  accident, stalled vehicles, illegal street parking, lane closure, or blockage within a link or within an intersection

- Excess demand

  due to special event, queues do not clear over several cycles

  left-turn pockets overflow over several cycles

  inadequate capacity due to inadequate effective green time available to a particular phase

- Detector malfunction

  system detectors

  traffic control detectors

Neural classifiers, as an ensemble of a great number of collectively interacting elements, are capable of storing representations of concepts and information as collective states. Therefore, different aspects of a pattern recognition problem can be expressed over a set of interconnections as weights in a distributed manner.

This research proposed the use of artificial neural networks in a modular architecture to detect the different types of operational problems listed above. The types of problem that can be detected depend on factors such as range of operating conditions, configuration of system detectors within the network, and block or link length. Neural network models were developed to demonstrate the feasibility of training and testing different architectures of neural network models as components of a modular architecture, with appropriate architecture for each sub-problem of pattern recognition. It was hypothesized that the performance of such a modular architecture would exceed that of any single

architecture applied to the detection of the different types of operational problems. A comparative analysis was carried out to study the performance of each type of model considered. Also included was a study of the effect flow levels and detector configurations have on the performance of the incident detection model. The results show that with the selection of a suitable architecture, the performance of the modular neural network classifiers developed based on data from loop detectors outperform other statistical techniques such as discriminant analysis. This is demonstrated by testing the detection of operational problems on street networks in the Cities of Los Angeles and Anaheim, California, using cyclic data collected from a microscopic-simulation, and the Urban Traffic Control System (UTCS) implemented in the field.

In this research we propose using a multiplicity of networks to take advantage of a modular architecture. As a result, a different network learns a different region of the input domain or class pattern by decomposing the problem at hand and splitting its input domain. A modular architecture was used to develop a hierarchy of neural nets to detect different types of problems under different operating conditions. From the tests performed for incident detection for arterials with various architectures, it was clear that the detection rate and the false alarm rates both increased simultaneously. Therefore, an attempt to increase the detection rate resulted in an increase in false alarm rate as well, i.e. the performance of one measure was inversely related to the other. This has also been found in incident detection for freeways (Ritchie and Cheu, 1990). Therefore, an attempt was made to train two separate neural networks, one to optimize the detection rate and another other to optimize the false alarm rate. It was also shown that, using two

differently trained neural networks, the false alarm rates could be lowered by the dual system of networks compared to a single network.

Incident detection systems, both for freeways and arterials, will operate in a traffic management center controlling and monitoring large area networks. Therefore, it is of utmost importance to keep the false alarm rates extremely low, so that when the neural network models are implemented in a real traffic management center, they will result in low false alarms. One of the main concerns of traffic engineers seeking an incident detection algorithm is not only a high detection rate, but perhaps of equal or greater importance is a low false alarm rate. As has been shown in the freeway case, even moderate false alarm rates can result in traffic engineers in a TMC ignoring real alarms. The dual neural network, composed of single networks, was trained separately to optimize the performance of detection rate and false alarm rate, and was jointly used to reduce the false alarm rates to an acceptable level for use by a traffic management center for large surface street networks.

The overall objective of this research was to:

- develop a methodology to automate the process of detecting operational problems on surface street networks
- extend the application of artificial neural networks to incident detection
- detect different types of problems
- test the robustness of the model developed by testing on different types of surface street networks and different detector configurations

1.3 Organization of Dissertation

The research effort is described in the following chapters:

Chapter 1 presents the problem addressed in this Dissertation, the approach proposed and the objectives of this research.

Chapter 2 presents a review of basic techniques applied to the problem of detecting non-recurring congestion or incidents on freeways, and also Dissertations the limited work done to develop a methodology for signalized surface street networks.

Chapter 3 identifies the problem addressed in this research as a pattern recognition problem, presents different approaches to solving pattern recognition problems, namely statistical and neural classifiers, the strengths and weaknesses of these approaches, and why neural network classifiers were proposed for the problem addressed in this research. Finally a neural network architecture was proposed to develop a comprehensive system to detect traffic operational problems for signalized arterials.

Chapter 4 describes the field and simulated data collected to develop and test the performance measures of the models developed for this research. The micro-simulator used, its calibration, the city networks represented, and the experiments designed are presented in this chapter.

Chapter 5 presents the model development, input feature selection, model structure, training and testing of the proposed model for different types of traffic operational problems under different operating conditions, and detector configurations.

Chapter 6 evaluates the results of the different neural classifiers, a statistical classifier, and the modular architecture of neural classifiers.

Chapter 7 discusses the findings of this research, and future direction of research in this area.

1. Incident Detection Approaches

1.1 Introduction

Detecting incidents on either a freeway section or a surface street is a pattern recognition, or more specifically a classification, problem. Algorithms have been developed for incident detection using various techniques. They can be classified as pattern recognition, pattern matching techniques or comparative algorithms, statistically-based algorithms, and traffic flow modeling-based approaches.

There are basically two approaches: one uses the notion of trying to find similarities and the other estimating beliefs/probabilities. Attempts to classify incident and non-incident data where the emphasis was on trying to determine the `similarities' included decision tree techniques, and time series or filtering techniques. These techniques ultimately rely on means to determine how close the traffic parameters are to some 'normal' values or predicted values using calibrated thresholds, determined differently by different techniques.

1.2 Basic Approaches to Incident Detection

1.2.1 Pattern Recognition-Based Algorithms

Pattern matching algorithms based on decision trees for freeway incident detection were developed by Payne and Tignor (1978), and were later developed by others (ref) as a

series of algorithms. They are based on decision trees to detect incidents from traffic parameters. These algorithms, better known as the California Algorithms, are based on the pattern of traffic when an incident occurs. When an incident occurs, congestion builds upstream of the incident - thus causing an increase in occupancy upstream, and decrease in occupancy downstream. But this difference can be also be caused by a bottleneck. Therefore the algorithm is also used to distinguish a bottleneck from an incident.

$$Occ_u(t) - Occ_d(t) \geq K_1 \qquad \textit{Eq. 1-1}$$

$$Occ_u(t) - Occ_d(t) \geq K_2 \qquad \textit{Eq. 1-2}$$

$$\frac{Occ_d(t-2) - Occ_d(t)}{Occ_d(t-2)} \geq K_3 \qquad \text{Eq. 1-3}$$

where,

Occ_u = upstream occupancy for time t (%)

Occ_d = downstream occupancy for time t (%)

K_1, K_2, K_3 thresholds

The first two tests (Eq. 2-1, Eq. 2-2) were used to compare the absolute difference in occupancy and the relative differences against thresholds. The third test (Eq. 2-3) determines whether the difference is due to a bottleneck or recurring congestion. Various versions of the algorithm have been developed based on this version. Currently, version

8 of this algorithm is being used for freeways in Los Angeles using 30 second occupancy values.

1.2.2 Time-Series Methods

Algorithms were also developed based on statistical forecasting of traffic behavior by time series algorithms (Cook and Cleveland 1974, Dudek and Messer, 1974; Ahmed and Cook, 1982). These time-series based methods provide a means of forecasting short term traffic behavior. Significant deviations from observed and estimated values of traffic parameters detect an incident.

1.2.3 Bayesian Approach

Levin and Krause (1978) have also used the Bayesian approach to classify incident and non-incident data. This algorithm uses the ratio of the difference between upstream and downstream one minute occupancies and also uses historical incident data. It is based on mathematical expressions derived from the ratio of distribution of incident and incident-free conditional distributions of incidents, given traffic features; and the probability of the occurrence of an incident at a particular location and time period. This algorithm performs better than the California Algorithm, but has a high mean time to detect.

1.2.4 Catastrophe Theory-Based Algorithm

The McMaster Algorithm (Persaud and Hall, 1989) is based on applying catastrophe theory to the two dimensional analysis of traffic flow and occupancy data, by separating

the areas corresponding to different states of traffic conditions. When specific changes of traffic states are observed over a period of time, an incident is detected.

1.3 Surface Street Incident Detection

Few attempts have been reported in the literature for surface street incident detection. One is based on decision trees (Han and May, 1989) another applies time-series technique to data from an urban traffic control system and simulated data for an isolated intersection (Bell and Thancanamootoo, 1988); a knowledge-based approach uses video image processing data (Sellam, Boulmakoul, and Pierrelee, 1991); and a discriminant analysis method uses traffic parameters from loop data and detector configuration data (Chen, and Chang, 1993). A description of each of these attempts is presented next.

1.3.1 Time-Series Based Algorithm

A time series approach was used by Bell and Thancanamotoo (1988) in an effort to perform incident detection. Incidents were defined as an unexpected, non-recurrent, longer term loss of capacity at a critical location. The key variables (determined by the detector type) at each detector site were collected on a cyclic basis. When the traffic condition remained normal, the mean and variance of traffic parameters were updated or estimated each cycle by exponential smoothing according to Eq. 2-4 and Eq. 2-5. Abnormal conditions were identified when the estimated key variable values were outside the range of an upper and lower bound as computed in Eq. 2-6, Eq. 2-7, and Eq.

2-8 where the bounds were established in terms of the smoothed mean and variance (Eq. 2-9 and Eq. 2-10).

$$\hat{F}(t) = 0.8\hat{F}(t-1) + 0.2F(t) \qquad \textit{Eq. 1-4}$$

$$\hat{O}(t) = 0.8\hat{O}(t-1) + 0.2O(t) \qquad \textit{Eq. 1-5}$$

$$F(t) < \hat{F}(t) - \alpha_1\hat{\sigma}_F(t) \qquad \textit{Eq. 1-6}$$

$$O(t) > \hat{O}(t) + \alpha_2\hat{\sigma}_o(t) \qquad \textit{Eq. 1-7}$$

$$O(t) < \hat{O}(t) - \alpha_3\hat{\sigma}_o(t) \qquad \textit{Eq. 1-8}$$

$$\hat{\sigma}_F(t) = 0.1(F(t-1) - \hat{F}(t-1))^2 + 0.9\hat{\sigma}_F(t-1) \qquad \textit{Eq. 1-9}$$

$$\hat{\sigma}_o(t) = 0.1(O(t-1) - \hat{O}(t-1))^2 + 0.9\hat{\sigma}_o(t-1) \qquad \textit{Eq. 1-10}$$

$F(t)$ and $O(t)$ were observed cyclic flow and occupancy, $\hat{F}(t)$ and $\hat{O}(t)$ were estimated flow and occupancy, and $\hat{\sigma}_F(t)$ and $\hat{\sigma}_O(t)$ were estimated standard deviations of flow and occupancy. The thresholds respond to changes in level of traffic due to exponential smoothing. The bounds were established based on the smoothed mean and variance. Whenever an incident was suspected, the mean and variance were frozen to avoid the incident affecting the mean and the variance computation. But if after the freezing of the

mean and variance for 5 consecutive cycles, an incident was not confirmed, the values were reset to the most current and smoothing commenced.

When the upper bound of occupancy was exceeded, an incident was suspected upstream, and the downstream occupancy was checked. If that too was found to exceed the lower bound then an incident was confirmed. On the other hand, if the lower bound was exceeded, then an upstream detector was checked for upper bound infringement to confirm an incident.

SCOOT data from the Traffic Management Division of TRRL in London for an isolated T-intersection were collected. There was an incident where a vehicle was parked close to the stop line for an hour. Data from the Traffic Control System Unit using the SCOOT system for 2 hours. This set consisted of data also from a T intersection for both direction of traffic. In this case the incidents were right turning vehicles blocking the traffic. Both incidents were detected using the algorithm developed. SCOOT data were collected for Middlesbrough, over a 2-hour period, which covered the evening peak period for 2 days. The incident detection algorithm developed produced no false alarms for this data set.

This approach was used by researchers in the DRIVE project MONICA (Monitoring Incidents and Congestion Automatically). Bretherton and Bowen (1991) report their work with the algorithm developed by Bell and Thancanamootoo which extended it to an arterial, using detector data from adjacent intersections. Data were collected using a

system developed for the London UTCS to receive, process and store traffic information produced from SCOOT. Data were collected over a three hour period for two consecutive links, where for an hour there was a lane blocking incident. The paper reports of a field testing to take place, but no follow-up literature was available reporting results.

1.3.2 Knowledge-Based System Using Video Image Processing

Sellam, Boulmakoul, and Pierrelee (1991) developed a knowledge-based system using video image processing for incident detection for signalized intersection. Their paper presents the general architecture of a system developed as part of the DRIVE project INVAID. It consists three Units: the Image Processing Unit (IPU) which outputs a binary image of the junctions or streets, a Measurements Processing Unit (MPU) which processes the binary image data to compute indicators of traffic and a Diagnosis Processing Unit (DPU) which diagnoses a problem and if necessary make requests to the MPU. The incident detection is based on the binary output of the digitized image - each black pixel on the source image represents a "moving vehicle" and a white image corresponds to the background. A user-specified parameter determines the time interval until which a stopping vehicle would be considered as a moving vehicle and after which it would be considered a parked vehicle. This results in a spatial detection algorithm as opposed to a vehicle detection.

1.3.3 Decision Tree-Based Approach

Shortly after Thancanomotoo's work was reported, Han and May (1988,1989) reported their attempt to develop an incident detection algorithm for surface streets. They selected a downtown area in Los Angeles under ATSAC (Automated Surveillance and Control System) with signalized surface streets of short blocks (400-500 feet) and detectors in all lanes. Detector data collected were smoothed.

The algorithm first uses the smoothed data to detect abnormal detector data patterns. Based on historical flow, occupancy and speed data for statistical ranges of 1 and 99 percentiles, comparisons are made.

After a check of detector malfunction, the algorithm proceeds to determine whether either an Impending Saturation Occupancy or an Impending Congestion Occupancy is exceeded. The flows are also checked against Medium and Low Flow thresholds. Thresholds of 300 and 500 for flow, and 30% and 40% for occupancy, are used. These values are determined for the test, in an attempt to minimize the false alarm rate. These thresholds are therefore time and detector dependent. One minute and three minute smoothed data were used to determine the thresholds. Traffic conditions on a street were classified into one of six states, depending on the occupancy and volume. Based also on the condition of adjacent lanes and downstream streets, classification as a lane blockage, approach blockage or arterial blockage was also made.

The algorithm was implemented as a system in C. It was tested off-line for a section of a street in Los Angeles (Washington Blvd.) near the Coliseum. This testbed had 20 detectors. In the early stages of development, runs were made to determine the thresholds using one minute and three minute smoothed data. In 1989, Han and May reported the development of TOPDOG, developed in TurboProlog, based on the algorithm described. They also reported using 50 minute data from Venice Blvd., Los Angeles. The system is still in its initial stages of testing as a demonstration prototype. Still further work is required to test whether a global set of thresholds may be determined for a detector configuration, as opposed to using time and detector dependent thresholds.

1.3.4 Discriminant Analysis-Based Approach

Discriminant analysis has been proposed for incident detection on surface streets by Chen and Chang (1993). This paper very briefly presents an incident detection algorithm for surface streets as part of a 3-module system - a dynamic traffic flow prediction model, an incident detection model, and an incident monitoring module. The paper presents the overall architecture of an incident detection system. According to the architecture presented, the flow model captures the dynamics of the traffic and thus predicts the flow conditions; this is compared to the real-time condition and forms the basis of the detection module. The paper presents the detection portion of this system:

$$d_{1-\text{lane blockage}} = -106.7 + 3.54\,ASPDUS + 7.22\,ASPDDS + 0.66\,AIFLDS + 2.21\,AOCUS$$
$$- 53.70\left(\frac{DFSPD}{DTSPACE}\right) + 1.69\,DFOC + 0.88\,TRUCK + 1.73\,BLRATO$$
$$+ 1.06\,DSP12 + 2.26\,DSP23$$

$$d_{2-\text{lane blockage}} = -131.26 + 3.39\,ASPDUS + 6.80\,ASPDDS + 0.10\,AIFLDS + 2.17\,AOCUS$$
$$- 156.88\left(\frac{DFSPD}{DTSPACE}\right) + 1.49\,DFOC + 0.05\,TRUCK + 1.58\,BLRATO$$
$$+ 1.89\,DSP12 + 2.04\,DSP23$$

where,

ASPDUS, *ASPDDS*	average upstream and downstream speed
AIFLD	average downstream flows
AOCUS	average upstream occupancy
DFSPD	upstream and downstream speed difference
DFOC	upstream and downstream occupancy difference
DSP12	speed difference of lane 1 and 2
DSP23	speed difference of lane 2 and 3
BLRATO	fraction of the blocked area of a link
TRUCK	composition of heavy vehicles
DTSPACE	detector spacing

This method is based on a set of multivariate discriminant functions. The paper presenting this method reports a misclassification rate of 13.22% and a variance of 0.59 for one and two-lane blockages classified in an experimental design of a 3-lane arterial

segment. The paper does not present the configuration of the network simulated, and the details of design of the experiments conducted, such as the location of the blockages, and whether the blockages were partial or complete lane closures. NETSIM, a microscopic simulation was used for the experiments, where only one lane blockages (partial blockage as stalled vehicle) can be simulated. The use of the variable 'fraction of the blocked area of a link' as reported in the paper suggests use of the 'lane closure' simulation feature of NETSIM that requires specifying the percent of time the lane closure is simulated, not lane blockages. Also, no information was available on how the detector data collected was available as the current version of NETSIM does not simulate surveillance detectors.

All of the incident detection methods described here are in their early stages of development, unlike the freeway incident detection approaches. It may be mentioned here that none (except the Chen and Chang paper) report any performance measures for the algorithms presented. The discriminant analysis based algorithm reports only misclassification rate, but no false alarm rates nor times to detection.

1. Pattern Recognition and Neural Networks

1.1 Introduction

The task of determining a procedure to use currently available information on an object or an event to assign it to a prespecified set of categories or classes is termed pattern classification or pattern recognition. A body of work has developed out of the extensive study of pattern recognition problems which has led to the development of mathematical models to design classifiers as shown in Figure 1-1. These models use a set of features of an object or an event and describe a relationship between these features (inputs) and its class pattern.

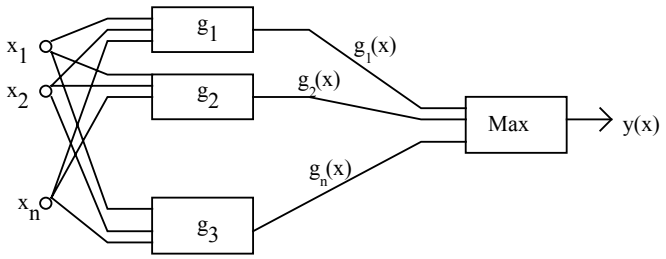

Figure 1-1. Pattern Classifier

(Duda and Hart, 1973)

In the literature there are a few types of pattern recognition techniques. They are mostly based on statistical, machine learning or neural network-based approaches. In this research, statistical and neural network approaches were considered and evaluated, and the discussion will be limited to these two.

1.2 Statistical Approaches to Pattern Recognition

In this approach, the problem of pattern recognition is considered a problem of estimating density functions in a high dimensional space and dividing the high-dimensional space into the regions of patterns or classes. The input features are considered realizations of random vectors, where the conditional density functions depend on the class pattern, and the density function may be known or assumed. The performance of a classifier can be analyzed for a given distribution of input vectors. In designing a classifier, the conditional density form may be known or assumed and the functional form of the classifier or discriminant can also be assumed to be linear, quadratic or piecewise linear. The best classifier for a given distribution is based on Bayes decision theory and minimizes the probability of classification error; it is also considered the optimal classifier. When the density function is assumed, the parameters of the function need to be estimated using parametric techniques, and when the density function is not known nonparametric techniques are used. For problems where the data do not fit the common density functions, non-parametric techniques are applied. However, nonparametric techniques are normally used for off-line analysis because of the limitations in performance, storage requirement, speed and complexity of the algorithms. Therefore,

this research, where a methodology is required to perform on-line, real-time analysis, statistical nonparametric methods are not discussed further.

In general, it is also known that it is easier to design a classifier for an input feature vector of lower dimensionality. Therefore, techniques are also used in pattern recognition that reduce the dimensionality of a given input vector to perform feature extraction, which then forms the basis of a linear classifier.

1.2.1 Bayesian Classifier

One of the fundamental approaches to pattern recognition is based on Bayesian decision theory, and is expressed in terms of probability structures. When the distributions of the input feature random vectors are given, it can be shown that the Bayesian classifier is the best classifier which minimizes the probability of classification error (Duda and Hart, 1973).

1.2.2 Discriminant Functions

Even when the probability distribution of the input feature vectors is given, implementation of the optimal Bayes classifier is often difficult when the dimensionality of the input feature vector is high. In such cases, another statistical classifier is often used, - discriminant analysis, where the mathematical forms of the discriminant functions are known (linear or quadratic classifiers).

Classification into groups or pattern classes is based on differences in the characteristics of the features of objects that come from the different classes. A good classification rule for discriminating between the classes minimizes the misclassification rate, the prior probabilities of occurrence of each class, and the cost of classification. Therefore, discriminant functions are to take these factors into consideration.

Discriminant function (DF) procedures are based on normal populations. Let $f_1(x)$ and $f_2(x)$ be multivariate normal densities, with mean vector and covariance matrix μ_1 and Σ_1, μ_2 and Σ_2 respectively. When $\Sigma_1=\Sigma_2$, the classification rule that minimizes the expected cost of misclassification is,

For class 1:

$$(\mu_1-\mu_2)'\Sigma^{-1}x_0-\frac{1}{2}(\mu_1-\mu_2)'\Sigma^{-1}(\mu_1+\mu_2)\geq \ln\left(\frac{p_2}{p_1}\right) \qquad \text{Eq. 1-1}$$

and for class 2 otherwise.

where,

p_1, p_2 = prior probabilities of belonging to class 1, and class 2, respectively

The classification rule in this case is linear, but when $\Sigma_1\neq\Sigma_2$, that is the covariance structure is different, then the discriminant function become,

For class 1:

$$-\frac{1}{2}x'\left(\Sigma_1^{-1}-\Sigma_2^{-1}\right)x+\left(\mu_1'\Sigma_1^{-1}-\mu_2'\Sigma_1^{-1}\right)x-k\geq \ln\left(\frac{p_2}{p_1}\right) \qquad \textit{Eq. 1-2}$$

and for class 2 otherwise.

where,

$$k = \frac{1}{2}\ln\left(\frac{|\Sigma_1|}{|\Sigma_2|}\right) + \frac{1}{2}\left(\mu_1'\Sigma_1^{-1}\mu_1 - \mu_2'\Sigma_2^{-1}\mu_2\right)$$

In this case, the classification regions are defined as quadratic functions of x. Therefore, in using discriminant functions as classifiers, assumptions of normality are made for the multivariate density functions, and linear and quadratic classifiers arise based on the structure of the covariances. If the data are not multivariate normal, the data may be transformed to variables that are more normal, and the linear or quadratic discriminant classifier can be used to determine the appropriateness of a particular classifier. Or, when appropriate transformations of the data can not be formed, the linear or quadratic classifying rule may be applied to check the performance of the classifier (Johnson and Wichern, 1992) and to determine whether linear or quadratic decision surfaces can perform the classification reasonably well for the particular pattern recognition problem.

1.2.2.1 Fisher Discriminant Function

In this linear discrimination rule, the assumption of normality for the multivariate data is not made. However implicitly the population covariance matrices are assumed to be equal. In this case, the rule maximizes the differences between the classes by maximizing the ratio of the squared distance between the sample means and the sample variance. The separation distance is expressed not in terms of the original input variables *x*, but a set of transformed variables *y*. The *x* variables are transformed to *y* by taking a

linear combination of *x*'s that maximize the separation distance in terms of *y*. The Fisher discriminant function becomes, :

For class 1 if:

$$y = (\bar{x}_1 - \bar{x}_2)' \Sigma^{-1}_{pooled} x \geq \frac{1}{2}(\bar{x}_1 - \bar{x}_2)' \Sigma^{-1}_{pooled} (\bar{x}_1 - \bar{x}_2) \qquad \textit{Eq. 1-3}$$

and to class 2 otherwise.

For the Fisher discriminant function, the two classes are assumed to have a common covariance matrix. From the discriminant developed, the maximum relative separation distance can also be calculated and the significance of the difference in means can be computed. A significant difference in means for the classes does not imply that the classifier developed will perform good classification. If good separation between the means does not exist, then the classification need not be performed. But if the difference in means is significant, then other methods of testing the validity of classification procedure is used.

1.3 Artificial Neural Networks

Classifiers based on Bayes decision theory can be shown to be optimal classifiers. But for Bayesian classifiers, the conditional probability density function for each class must be known. Even when the densities are known, they may be difficult to implement. A classifier is designed by assuming the mathematical form of the classifier (linear,

quadratic or piecewise linear) and the parameters have to be estimated. But the performance of the classifiers depend on certain conditions of the density functions and their covariance structures. It may be mentioned that the Bayesian linear classifier is optimal only when the distribution is normal and the covariances are equal. When the assumption of equality of the covariances is inappropriate, the Bayesian classifier is not optimum. For unequal covariances or non-normal distributions, quadratic discriminants or other types of classifiers may be developed. But as pointed out, often the robustness of linear classification is preferred to the performance of the more complex classifiers. Therefore, for most pattern recognition problems, linear classifiers are initially designed, and then the performance evaluated.

In recent years, artificial neural networks have been applied to a variety of pattern recognition problems and have clearly emerged as one that has outperformed some statistically based techniques described in the previous section. But as pointed out in the previous section, the performance of statistical classifiers depends on how well the assumptions about the density functions fit the data they describe, and also on the appropriateness of the functional form of the discriminant function. These factors are application-specific. When the restrictive assumptions are violated for a particular application, the classifier is no longer best or optimum. Another set of classifiers that address theses types of problems are neural network-based classifiers.

Artificial neural network architectures that were initially developed and successfully applied, were based on the original perceptron [Rosenblatt, 1958]. They are parallel

distributed processing information structures that combine computational and knowledge representation methods. An artificial neural network consists of many processing elements (PE's) that are massively connected with each other. The processing elements can be arranged in layers, with an external layer receiving input from data sources, which is passed on to other processing elements through interconnections, and on to an output layer of processing elements. Since these structures are distributed in nature, they are known to be robust, efficient, and also have the capability to capture highly non-linear mappings between inputs and outputs.

Processing elements each receive inputs from external data sources or other processing elements and pass through these summation function, a transfer or activation function, as shown in Figure 1-2. The method of obtaining the weights to perform the desired mapping is termed as learning. The various types of training methods are described in the next section.

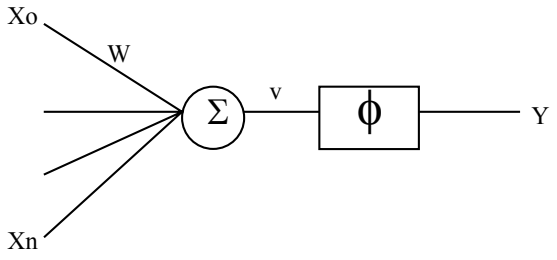

Figure 1-2. A Processing Element

1.3.1 Learning Schemes

1.3.1.1 Supervised Learning

Supervised learning occurs when the network parameters are adapted under the combined influence of the error signal and the input vector. This adjustment is often made by an iterative procedure until the output produced follows the target signal closely in some statistical terms according to either a least mean square or steepest descent algorithm using the instantaneous estimate of the gradient of the error surface.

1.3.2 Unsupervised learning

Unsupervised learning is controlled by a teacher or an external target signal and the performance of the network weight is updated based on the ability to follow the target signal. For supervised learning, learning is carried out not in terms of a cost function, which is expressed in terms of an error signal, but rather based on a target independent measure. Network parameters are adjusted based on an independent measure of performance which captures statistical regularities of the input vector and forms classes of patterns automatically.

1.3.3 Competitive Learning

This occurs when the processing units compete amongst themselves to respond to a stimulus or input. In Hebbian learning the units produce outputs simultaneously, but in competitive learning only one unit responds at a time and is therefore often referred to as the winner-take-all method. For a collection of processing units, certain units specialize on certain groups or classes of patterns and therefore work well to detect certain features of an input set.

Based on this learning scheme where the weights are normalized and therefore the input vectors also properly scaled lie in an N-dimensional unit hypersphere. the learning moves the weight vector to the center of gravity of the cluster it discovers.

1.4 Artificial Neural Networks for Pattern Recognition

Artificial neural networks applied to pattern recognition problems have clearly emerged as a major contender to other statistical-based approaches. The applicability of one approach over the other is of course application specific. Statistical-based work in pattern recognition is based on discriminant analysis or on a class of models that attempt to provide an estimate of the joint distribution of the features within each class. There are approaches that are characterized by having an explicit underlying probability model of being in each class. The main advantages of applying neural network techniques are described as follows,

- Only weak assumptions need to be made about the data set as opposed to statistically-based approaches where more restrictive assumptions of the underlying distributions are made to obtain the optimal classifiers.

- Classifiers based on Bayes decision theory can be shown to be optimal classifiers. But for Bayesian classifiers, the conditional probability density function for each class must be known. Even when the densities are known, they may be difficult to implement. A classifier is designed by assuming the mathematical form of the classifier (linear, quadratic or piecewise linear) and the parameters have to be estimated. But the performance of the classifiers depend on certain conditions of the density functions and their covariance structures. It may be mentioned that the Bayesian linear classifier is optimal only when the distribution is normal and the covariances are equal. Based on the application, when the equal covariance assumption is inappropriate, the Bayesian classifier is not optimum.

- Even though there are nonparametric statistical techniques that can be applied to pattern recognition to avoid making assumptions about the distribution of input features, these techniques are difficult to apply to on-line applications as they are computationally intensive and require extensive amounts of storage space

- Nonparametric techniques also suffer from the 'curse of dimensionality', where although with enough samples, convergence to an arbitrary complicated unknown density function is assured, the number of samples required may be very large. The demand for a large number of samples grows exponentially with the dimensionality of the input feature space. This limitation severely restricts the practical application of nonparametric techniques. Work with MLF networks (Baron 1991, 1992) shows that the rate of convergence expressed as a function of the training set size N is of the order $(1/N)^{1/2}$ (times a logarithmic factor). This makes the application of neural network techniques feasible for problems that require on-line implementation.

- Since the relationships in ANN are expressed as a set of interconnections, they are distributed in nature and thereby more robust and computationally efficient

1.5 Artificial Neural Network Architectures

1.5.1 Multi-layer Feed Forward Neural Network

The multilayer feedforward neural network consists of an input layer, one or more non-linear hidden layers, and an output layer. It employs an error correcting back-propagation algorithm for training. In the case of backpropagation, there are two distinct

phases - a forward pass and a backward pass. In the forward pass the input is propagated through layers of processing units, and in the backward pass, the errors computed are propagated backwards with parameters that minimize the mean square errors (Rumelhart and McClelland, 1986).

The model of each processing element or units includes a non-linear transfer function that is part of each processing element in the hidden layers and output layer. The hidden layers progressively extract more and more features from the input data. The layers are interconnected through a set of weights. Due to this distributed form of non-linear processing, these structures are able to produce highly non-linear mappings between inputs and outputs. The weights in these networks are adjusted according to the well known back-propagation algorithm that minimizes the mean square error between the outputs produced by the network and a set of desired outputs.

The error is computed as,

$$e_j(n) = d_j(n) - y\ (n) \quad \text{for the output unit j} \qquad \textit{Eq. 1-4}$$

The sum of the square of the errors is,

$$E(n) = \frac{1}{2}\sum_j e_j^2(n) \quad \text{for all output units j} \qquad \textit{Eq. 1-5}$$

and the mean error over N training patterns,

$$E_{mean} = \frac{1}{N}\sum_{n=1}^{N} E(n) \qquad \textit{Eq. 1-6}$$

The objective is to adjust the parameters or weights. The errors computed for every pattern are thus summed over the entire set of patterns to produce an estimate of the overall error, which is used as a measure to adjust the weights. Considering a typical processing unit as shown in Figure 3-3, the net output produced is,

$$v_j(n) = \sum_{i=0}^{p} w_{jii} y_i(n) \quad \text{for output unit j from all p inputs} \qquad \textit{Eq. 1-7}$$

According to the back-propagation algorithm, the weight adjustment DWji is proportional to the instantaneous gradient,

$$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = \frac{\partial \varepsilon(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \qquad \textit{Eq. 1-8}$$

where,

$$\frac{\partial \varepsilon(n)}{\partial e_j(n)} = e_j(n), \quad \frac{\partial e_j(n)}{\partial y_j(n)} = -1, \quad \frac{\partial y_j(n)}{\partial v_j(n)} = \varphi'(v_j(n)), \quad \frac{\partial v_j(n)}{\partial w_{ji}(n)} = \mathrm{y_i(n)} \qquad \textit{Eq. 1-9}$$

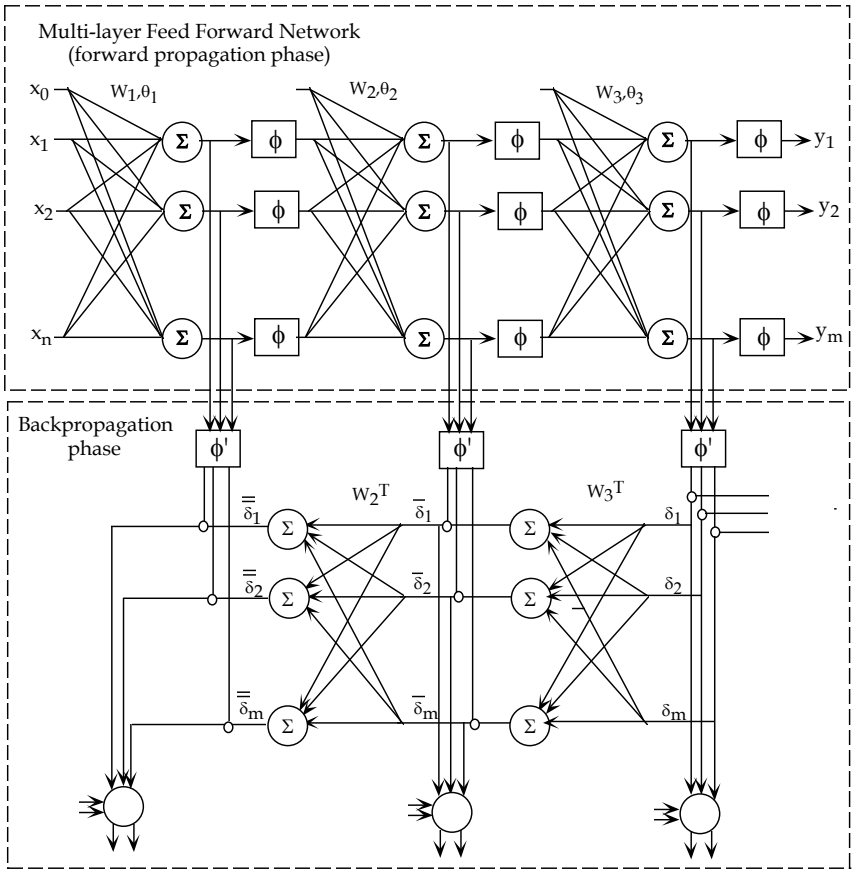

Figure 1-3. Multilayer Feedfroward Neural Network

Forward/Backward pass

(Haykin, 1992)

Therefore,

$$\Delta w_{ji}(n) = -\eta \frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = -\eta \delta_j(n) y_i(n) \qquad \textit{Eq. 1-10}$$

where,

$$\partial_j(n) = \frac{\partial \varepsilon(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = e_j(n) \varphi'(v_j(n)) \qquad \textit{Eq. 1-11}$$

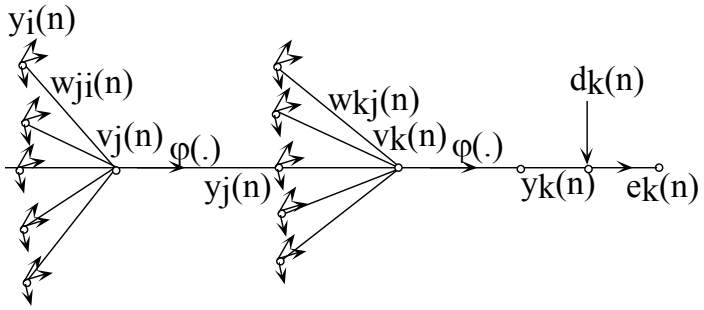

Figure 1-4.Hidden and Output Layer Processing Elements

For hidden layer units:

$$\partial_j(n) = \frac{\partial \varepsilon(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = \frac{\partial \varepsilon(n)}{\partial y_j(n)} \varphi'(v_j(n)) \quad \text{for hidden unit j} \qquad \textit{Eq. 1-12}$$

$$\frac{\partial \varepsilon(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)} \quad \text{for output unit k} \qquad \textit{Eq. 1-13}$$

where:

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi'(v_k(n)), \ \frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n)$$

and,

$$\frac{\partial \varepsilon(n)}{\partial y_j(n)} = -\sum_k e_k(n)\varphi'(v_k(n)w_{kj}(n) = -\sum_k \delta_k(n)w_{kj}(n)$$

where:

$$\partial_j(n) = \varphi'_j(v_j(n))\sum_k \partial_k(n)w_{kj}(n) \quad \text{for j hidden unit}$$

Therefore the learning rule may be summarized as:

$$\begin{pmatrix} Weight\ correction \\ \Delta W_{ji}(n) \end{pmatrix} = \begin{pmatrix} Learning\ parameter \\ \eta \end{pmatrix} . \begin{pmatrix} Local \\ gradient \\ \partial_j(n) \end{pmatrix} . \begin{pmatrix} Input\ of\ unit \\ y_i(n) \end{pmatrix} \qquad \textit{Eq. 1-14}$$

1.1.1 Projection Neural Network

Multilayer Feedforward (MLF) networks are capable of mapping any continuous bounded function of $_0$dimensions to $_0$outputs by dividing the input space into regions using hyperplanes. The locations of the hyperplanes are determined by the weights and thresholds of the hidden layer nodes. Non-linear combination of these hyperplanes can bound regions by hyperplanes or curved surfaces, either open or closed. But to develop these complex boundaries more regions are required. For a large set of N, large number of hidden layer nodes, therefore large networks are required. On the other hand, there is a different set of neural network architectures that train very fast but does not attempt to minimize error. Networks such as Kohonen networks, Radial Basis Functions, and a few others, attempt to place prototypes within closed decision boundaries around training data points and adjust to their parameters. These networks place hyperspheres around prototypes and adjust their radii.

A third set of neural networks have evolved that combine the ability to form closed boundaries and also perform error minimization. An examples of this type of network is the Projection network. This network can form both closed and open decision regions. Training will cause closed boundaries to open if required and vice versa. Details of this network will be presented in the next section. The advantage of this type of network lies in its ability to initialize rapidly to a good starting point, which substantially speeds up training.

The projection network is based on the concept of projecting the inputs to one higher dimension to form a hypersphere, where the weight vectors will also lie on this hypersphere. With a hidden layer, networks could be capable of forming either an open or closed region within the original input space. The idea of trying to project inputs to higher dimension and use the inner product of the input and the weights to determine the closeness of these two vectors has been used by other network architectures such as the Radial Basis Functions. But as in the case of Radial Basis Functions (RBF), the framework of clustering was used to form closed prototypes. But in the case of Projection Network, these boundaries are formed within the original framework of back propagation (discussed in the MLF section).

As mentioned earlier, the idea is to project the input vector from N dimensions onto a higher dimension (N+1) by transforming the input vector x to x', subject to |x|=R. Example of transformations is

$$x' = R\left(\frac{h}{\sqrt{h^2 + x^2}}, \frac{x}{\sqrt{h^2 + x^2}}\right)$$

Eq. ***Error! No text of specified style in document.****-1*

These projections then serve as inputs to a MLF network. Here h is the distance between the origin of the original input space and the (N+1) space. The component of the input vector lies along the extra dimension (N+1) and the others along the original dimension, therefore the weights that connect the (N+1) component to a hidden unit also lie on the extra dimension (N+1). These weights are also constrained to $|w|=R$. Figure Error! **No text of specified style in document.-1** shows the projection a 2-dimensional space to a

3-dimensional space. The vector X connects the origin of the 2-dimensional space, and the 3-dimensional vector X' connects the center of the sphere and the point X and extends the line to intersect the surface of the sphere. As X' is on N+1 dimensional space, W' that connects modified input X' to the hidden layer node is also 3-dimensional. The net input to the hidden layer node is,

$$\textit{Net input to the hidden layer node} = w'.x' - w_0 = \textit{constant} \qquad \textit{Eq. } \textbf{\textit{Error! No text of specified style in document.-2}}$$

As described for MLF, the output produced by input vector x and weights w with bias or threshold $w_{O,0}$ is passed through a non-linear function, often sigmoidal. The threshold determines the location of the decision boundary formed and is proportional to the distance from the origin to the decision surface or hyperplane.

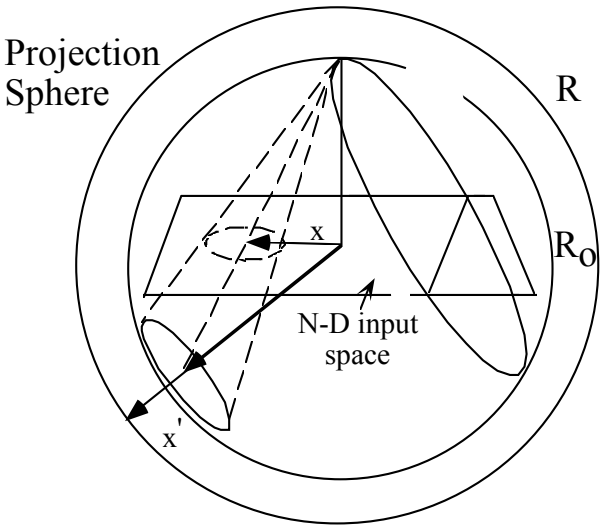

Figure ***Error! No text of specified style in document.****-1. Projection transformation and the formation of boundary surfaces*

(Wilensky and Manukian, 1992)

$_0$In (N+1) dimensions, each hidden layer node still draws a hyperplaner decision boundary that intersects with the hypersphere around X'. This is a circle around X' and the position of the intersection was determined by the threshold. Therefore, the projection of the surface resulting from the intersection back onto the original N-dimension space is a function of the threshold. As shown in Figure Error! **No text of specified style in document.-1**, if the threshold is large, the resulting intersection circle is small and lies on one side of the original 2-dimensional plane. If the threshold is small, the intersection circle approaches a great circle on the sphere and its projection back on the 2-dimensional plane is a curve or an open boundary or line. It is this ability of the projection network to form hyperplanner or hyperspherical prototypes that allows the Logicon network to b initialized rapidly to a good starting point. It is during learning that the closed boundaries can become open boundaries and vice versa, as the weights and the thresholds are adjusted.

The training of the projected weights is based on back-propagation, which minimizes the error by changing the weight vector in the direction of maximum error decrease. But in the case of the projection network, the weight vector has to be moved in the direction of maximum error decrease, but needs to be constrained to be tangent to the hypersphere surface in order to keep the weight vectors on the hypersphere. The change in weights is expressed as:

$$\delta w' = \frac{w'}{R} x \left(\frac{w'}{R} x \, \alpha \nabla e \right)$$

Eq. ***Error! No text of specified style in document.****-3*

where the *e* is the error between the desired output and the output produced by the network, Δe is the error gradient with respect to the weights and is the gain. The weights need to be normalized to have magnitude R to prevent the vectors from moving away from the hypersphere.

1.1.2 $_0$Modularity of Neural Network Models

A modular architecture allows decomposition and assignment of tasks to several modules. Therefore, separate architectures can be developed to each solve a sub-task with the best possible architecture, and the individual modules or building blocks may be combined to form a comprehensive system. The modules decompose the problem into two or more subsystems that operate on inputs without communicating with each other. The input units are mediated by an integrating unit that is not permitted to feed information back to the modules (Jacobs, Jordan, Nowlan, and Hinton, 1991). The modular architecture combines two learning schemes, supervised and competitive. The supervised learning scheme is used to train the different modules of the networks and a gating network operates in a competitive mode to assign different patterns of the task to a module through a mechanism that acts as a 'mediator'.

Neural networks are commonly designed at the level of processing elements or units which represent the finest level. Layers of processing elements are at a coarser level. Adding networks adds an even coarser level to the classification. However, there may be significant practical and theoretical advantages to be gained by considering modularity at

the network level. The advantages of modularizing are described in the following section.

1.1.2.1 Why Modularize ?

Proper Assignment of Tasks or Functions

Networks composing the modular architecture compete with each other and learn the training patterns. As a result, they learn different functions or tasks by partitioning the function into independent tasks and allocating a distinct network to learn each task. In addition, the architecture allows the allocation of a topologically appropriate network to each task. Often there is a natural way to decompose a complex task into a set of simple tasks. For example, the$_0$ non-linear function $y = |x|$ can be approximated either with a neural network using one layer of hidden units or by assigning two networks, one for each linear function when $x \geq 0$ *and* $x < 0$, each with a single linear unit without any hidden units, and by setting an appropriate switching or gating mechanism. This example shows that if the data supports a discontinuity in the function being described, then it may be more effective to fit separate models on both sides of the discontinuity. Similarly, if a function is simple in some region, then a global model could be fitted in that region rather than approximating the function locally with a large number of local models. By proper assignment of tasks, the network could be simplified in structure to perform the same set of tasks. Therefore, modular structures perform local generalization by learning patterns of a particular region. This ensures that the performance of a single module does not affect the other modules of the structure. In this example, the task of decomposition

simplified the structure by removing the need for hidden layers, thereby reducing computational speed.

Speed of Learning

As the example demonstrates, a modular network to train the two simpler networks would train faster in the absence of hidden layer processing elements. The modularization is able to take advantage of function decomposition and can also reduce the conflicting information that tends to retard learning. In the literature, this is termed 'crosstalk'; and may be spatial or temporal in nature (Jacobs, Jordon and Barto, 1986). In the MLF, crosstalk may occur when the backpropagation in MLF is applied to two or more outputs as shown in Figure 3-6.

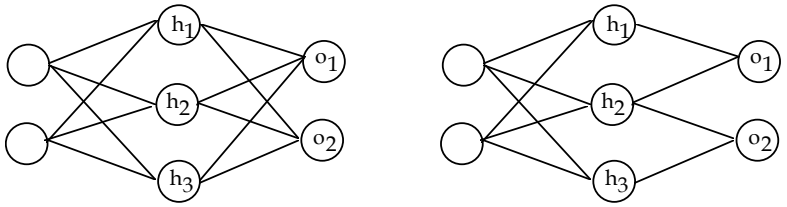

Figure ***Error! No text of specified style in document.****-2.* (a) MLF neural network (b) a modular equivalent

If the output of the hidden unit h_1 in Figure 3-6(a) produces positive weights to output units O_1 and O_2, and the first output O_1 is 'too large' and the second output O_2 is 'too small', then using the backpropagation derivative information will specify that for O_1 the hidden layer output should be smaller, while for the second output O_2 will suggest that the hidden layer output should be larger. This conflict in derivative information is

referred to as spatial crosstalk. Modular architectures, as noted by Plaut and Hinton (1987) are immune to spatial crosstalk.

Besides spatial crosstalk, there may be temporal crosstalk. For example, if a network is initially trained to learn a pattern, its hidden units become useful in performing that function. But when another pattern is trained, the performance on the first pattern may deteriorate. Often times with backpropagation training, this is overcome by adding more hidden layer units. But hidden units are added at a computational cost of speed and complexity. Use of modular networks may eliminate this need.

Representation

Modular structures can also provide a method of representation that is natural or easy to understand. The modules can be viewed as 'building blocks' for more comprehensive and complex tasks, where the idea is to 'divide and conquer'. This philosophy has long been used in computer science, and in numerical methods such as finite element analysis. The modular structure can provide a means of decomposition at a broad level which suppresses the activation of a large number of processing elements, while activating a smaller number of processing elements in a single module.

This structure also allows domain specific knowledge to be incorporated. For example, Jacobs, Jordan and Barto (1990) in their work decompose the task of 'what' and 'where' vision tasks to two different multi-layer feed forward neural networks. Knowledge can

also be incorporated into the design of a structure by deciding on how to divide the input information between the gating networks and the modules since there may be a natural context of division. Another way to incorporate domain specific knowledge is in the design when part of the functional properties may be known. For example, a linear or non-linear portion may be identified, and therefore different networks may be designed to possess different topologies, weights, activation functions, error functions, different input variables, etc.

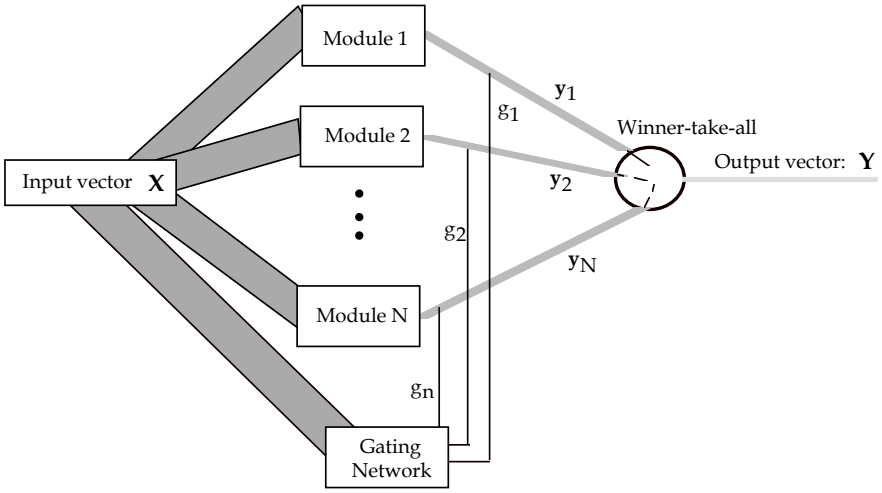

Figure 3-7. Modular Architecture

1.1.2.1.1 Algorithms

The problem of decomposing training cases into a set of sub-tasks was addressed by Hampshire and Weibel (1989) when these sub-tasks can naturally be identified beforehand. But Jacobs et al (1991) first presented the idea of a system that learns to allocate sub-tasks to different networks/experts or modules. The idea in this method is that a gating mechanism encourages one module or network to be assigned to a subtask, and that these modules would perform their tasks locally, decoupled from one another.

Therefore no crosstalk or interference between the weights occur as shown in Figure 3.5(b). Hamshire and Weibel (1989) in their work presented a method where the modules were not decoupled, since the final output was a linear combination of outputs produced by the individual modules. $_0$The interaction between modules caused some of the modules to be used for a subtask. But Jacobs and Jordan (1991) proposed another error function to encourage competition between modules using the gating network. The gating network makes a stochastic decision about which single expert to use on each occasion. Each module works as a feedforward neural network and all modules receive the same input and have the same number of outputs. The gating network is also a feedforward network and typically receives the same input as the other modules (Figure 3-7).

$$\mathbf{y} = \sum_{i=1}^{n} g_i \mathbf{y}_i \qquad \textit{Eq. 3-16}$$

During training, the weights of all the networks are modified simultaneously as in the multilayer feedforward network. The training of the modules and the gating network are based on error minimization of the error functions. The error function for the modules is,

$$J_y = \frac{1}{2}(\mathbf{y}^* - \mathbf{y})^T(\mathbf{y}^* - \mathbf{y}) \qquad \textit{Eq. } \textbf{\textit{Error! No text of specified style in document.-4}}$$

where, y* is the desired output, output of the system is y.

The error function of the gating mechanism is more elaborate. For each training pattern, one module comes closer to producing the desired output than the others. If for a given training pattern the systems performance is improved significantly than in the past, the weights of the gating networks are adjusted to make the output of the winner increase towards 1, and the outputs of the loosers towards 0. If the systems performance does not improve, the gating weights are adjusted to move all of the outputs toward some neutral value. The function that determines whether the performance of the model has improved is,

$$\bar{J}_y(t) = \alpha J_y(t) + (1-\alpha)\bar{J}_y(t-1)$$

Eq. ***Error! No text of specified style in document.****-5*

where, α, $0<\alpha<1$. This determines how rapidly the past values are forgotten by exponentially weighting the average of J_y. The binary variable λ_{WTA} and λ_{NT} indicate whether the performance has improved. That is,

$$\begin{aligned} &\textit{If} && J_y(t) < \gamma \bar{J}_y(t-1) \\ &\textit{Then} && \lambda_{WTA} = 1 \quad \textit{and} \quad \lambda_{NT} = 0 \\ &\textit{Else} && \lambda_{\mathrm{WTA}} = 0 \quad \textit{and} \quad \lambda_{NT} = 1 \end{aligned}$$

Eq. ***Error! No text of specified style in document.****-6*

where γ is a multiplicative factor that determines how much less the current error must be than the measure of the past.

If the architecture's performance improves significantly, the module whose output is closest to the desired output is determined. The error for the module is then defined as the sum of squared error between the module's output y_i, and the desired output y*. The error is then,

$$J_{yi} = \frac{1}{2}(\mathbf{y}^* - \mathbf{y}_i)^T(\mathbf{y}^* - \mathbf{y}_i)$$

Eq. ***Error! No text of specified style in document.****-7*

The module that wins is the module with the smallest error. If network or module I wins, then the desired value of the ith output of the gating network g_i is set to 1 and otherwise set to 0. If the systems performance does not improve significantly, that is when λ_{NT}=1, then the weights are adjusted so that the outputs of the gating network move towards a neutral value 1/n, where n is the number of modules. The gating networks error function is defined as,

$$J_G = \lambda_{WTA}\frac{1}{2}\sum_{i=1}^{n}(g_i^* - g_i)^2 + \lambda_{WTA}\frac{1}{2}(1-\sum_{i=1}^{n}g_i)^2 + \lambda_{WTA}\sum_{i=1}^{n}g_i(1-g_i) + \lambda_{NT}\frac{1}{2}\sum_{i=1}^{n}(\frac{1}{n}-g_i)^2$$

Eq.

Error! No text of specified style in document.*-8*

where, the first three terms of Eq. 22 contributes to the error when the performance of the system improves, and the fourth term contributes when the performance has not improved significantly. The first term is the sum of the sum of the squared error between the desired outputs and the actual outputs of the gating network. the second term takes its smallest value when the outputs of the gating network sum to one, the third term takes

its smallest value when the outputs of the gating network are binary valued, and the fourth term is the sum of the squared error between the neutral value and the actual outputs of the gating network. When the performance of the system does not improve significantly, the last term causes all of the outputs of the gating network to approach the neutral value of 1/n. This architecture was used to develop a modular neural network model to detect different types of operational problems on signalized surface street networks.

0

1. Data Collection

1.1 Introduction

To study the feasibility of applying neural network techniques to detecting operational problems for surface street networks, data from inductance loops were collected from selected field sites. Due to limited availability of field data, where the incident characteristics such as the time, duration and location of each incident were clearly identified, both simulated and field data were used. Initially, a microscopic simulator for surface street networks was calibrated to replicate loop data from the field. To develop and test neural network models, evaluate the performance of these models, and compare their performance to other models developed based on statistical classifiers, data were collected from the calibrated micro-simulator. Additional field data were collected to test the performance of the models developed.

The traffic management centers where data were collected use the Urban Control System (UTCS) software. The street networks have two different sets of loop detectors embedded in the pavement. One set controls the operations of the signals under actuated control; another set, the system loops, are normally located a few hundred feet upstream of the stop line to collect traffic surveillance data (volume, occupancy and speed data). The field data collected from system loops and simulated data from other locations were also used for this research.

Normally, the traffic management centers only collect 15-minute detector data from UTCS. For the purposes of this study, cyclic data were collected for all the intersections of the study areas in this research, and the models developed were trained and tested in an off-line mode. Two study networks were selected, one in the City of Anaheim, another in the City of Los Angeles. For both of these locations, field data were collected for conditions when traffic operational problems existed, and also when the traffic conditions were incident-free. These study networks were selected for their unique locations, and having several special event locations such as the Disneyland and the Anaheim Convention Center area, good detectorization, different detector configurations and network geometry. As mentioned earlier, both simulated and field data were used in this research.

In the next few sections, a detailed description of the study areas, the simulator, and the field data collected is presented.

1.2 Signalized Street Networks Selected as Study Areas

Two networks were selected for this research - one in Anaheim, California, controlled by the City of Anaheim's Traffic Management Center, and another in Los Angeles, California, controlled by the Los Angeles Automated Traffic Surveillance and Control System (ATSAC) Center. In both cases, the overall task of intersection or area control, data collection, and information reporting is done by the Urban Traffic Control System

(UTCS) software. Initially developed by the Federal Highway Administration (FHWA), it has been modified and enhanced by a consultant for both Los Angeles and Anaheim.

1.2.1 Traffic Control System

In the enhanced version of UTCS, communications to and from all of the intersections are done once per second. UTCS sends timing plan information to the traffic controllers and the controllers return equipment status and detector data. A front-end processor performs the task of controller communications, and a host computer does overall management. UTCS does vehicle detection and reports the data to the TMC. The vehicle detection is performed by the inductive loop detectors that are installed 100-400 ft from the intersection stopline. UTCS collects detector volume and occupancy data and averages it over a user defined interval (cyclic or minute) to produce Detector Analysis Reports (DAN). 15-minute detector data are normally collected seven days a week, and forms a history of the detector data, in a Detector History Report (DHS).

1.2.2 Description of Study Networks

1.2.2.1 Los Angeles Network

A part of the city of Los Angeles city network on Washington Blvd. between Hauser Blvd. and Soto St. (Figure 1-1) was selected for this research. The Santa Monica Freeway (10) runs east-west south of Washington Blvd., and the Harbor Freeway (110) runs north-south east of Washington Blvd. These streets have detectors on through lanes, and in left turn pockets, and operate on fully-actuated mode using pre-stored base timing plans in the database, based on time of day. Sections of this area can also be monitored by a video camera. The block length on these streets varies from 400 to 2000 ft. Most have two through lanes, with a left turn pocket. Depending on the block length, the system loops are located 98 to 350 ft. upstream of the intersection stopline. Data were collected from this network during the evening peak and mid-day peak period, both without incidents or operational problems and with operational problems such as lane blocking incidents, and detectors malfunctioning. The cycle length varied from 90 to 120 seconds.

Figure 1-1. Los Angeles Network

1.2.2.2 Anaheim Network

The network selected is adjacent to Disneyland, the Anaheim Convention Center, and the Interstate 5 Freeway. It includes Katella Blvd. from Haster St. on the east to Walnut St. on the west (Figure 1-2). Traffic signals in this network are centrally controlled by Urban Traffic Control System (UTCS) software and are managed by the City of Anaheim's Traffic Management Center (TMC). The intersections operate according to a background coordinated timing plan, with traffic-responsive control at each intersection. Some intersections are also equipped with video cameras. The network has all three lanes detectorized for almost all the streets. The block length on this network varies from 700-1200 ft. Since it is located near two major activity centers - Disneyland and the Convention Center, this network was selected to collect data not only for lane blocking problems but also for special event conditions. These activity centers on certain days have 50,000 to 70,000 people attending, generating extremely high volumes of traffic. Timing plans are assigned based on the expected number of attendees. But in certain locations, the special event timing plans are not adequate to cope with the demand either due to more than expected demand or due to prevailing conditions. These conditions need to be detected by an algorithm, and therefore data were collected for these conditions to train and test the performance of the neural network models.

Figure 1-2. Anaheim Network

1.3 Field Data Collected

For the networks selected, 15-minute detector volumes were plotted to select different time periods to simulate. During the PM peak period (16:00-19:00) flow was heaviest for both networks. During the AM period (7:30-9:00) for the Anaheim network and Mid-day for the LA network, there was medium flow. For Anaheim, one hour of each of the AM (7:30-8:30) and the PM (17:30-18:30) periods was simulated for calibration purposes. Similar plots were made for the LA network, and one hour of the PM peak period (18:00-19:00) was selected. Table 1-1 and Table 1-2 describes the data collected from both the Anaheim and Los Angeles network respectively.

Table 1-1. Data Collected from the Anaheim Network

Data Set #	Date	Time	Description
1	Aug., Sept., Oct., 1993	AM Peak	Incident-free
2	2 days each in Aug. - Oct. 1993	AM peak	Special event: Convention Center
3	Aug.-Oct. 1993	AM peak	Detector malfunction

Table 1-2. Data Collected from the LA Network

Data Set #	Date	Time	Description
1	October, 1993	PM peak	Incident-free
2	October, 1993	Mid-day peak	Incident-free
3	October, 1993	PM peak	Lane-blocking incident: stalled vehicle close to the intersection
4	October, 1993	PM peak	Lane blocking incident: stalled vehicle mid-block
5	October, 1993	PM peak	Lane blocking incident: stalled vehicle at the upstream end
6	October, 1993	PM peak	Detector malfunction
7	October, 1993	Mid-day peak	Lane-blocking incident: stalled vehicle
8	October, November, December, 1993	Mid-day peak	Detector malfunction

For the period selected, data were collected using the UTCS report generation capability. The UTCS 'Detector Data Analysis (DAN)' and '15-Minute Detector History Report (DHS) were used to collect unsmoothed cyclic volume and occupancy for the detectors in place at each intersection for the network. For some intersections, during the period data were collected, there were some detectors malfunctioning. For these approaches to the intersections, the video cameras available were used to record the intersection operations

at the TMC. The video tapes were later analyzed to gather volume data to estimate entry flows to the network simulated using NETSIM.

Video data were also used to compute the average start-up lost time and the queue discharge characteristics. For the intersections, one hour of operation was video taped using the Anaheim TMC cameras mounted on light poles. These cameras allowed all three lanes for one sync phase (movement 2) approach to be recorded in one frame. Therefore the headways of the vehicles as they discharged from the signalized intersection and the start-up lost time could be measured for the first vehicles as well as the other queued vehicles.

1.4 Microscopic Simulation

NETSIM is a microscopic, interval-based, stochastic simulation model of traffic operations on urban street networks. It was initially developed in the 1970's for the Federal Highway Administration (Peat et al, 1971) and has been subsequently modified. The model was later integrated within the TRAF simulation system (FHWA, 1992). The simulation model has been extensively used for various studies (Williams, Mahmassani, and Herman, 1984; Luk, 1984; Williams, 1986; FHWA, 1986; Serano, 1990; Sulzberg, 1991; Eshraghnia, 1992).

NETSIM describes the dynamics of traffic operations on urban streets. Vehicles are represented individually and are moved every second, and control devices and events are

updated every second. Each vehicle belongs to one of four categories (auto, carpool, truck, bus) and to one of 16 types, based on differing operational and performance characteristics. In addition, driver characteristics may also be specified. Vehicles are moved based on a car-following logic, in response to traffic control devices and demands. Each time a vehicle is moved, its position on the link and its relationship to other vehicles nearby is recalculated. Actuated control and interactions between vehicles are explicitly modeled. Lane blockages can also be simulated where vehicles respond by switching lanes where sufficient gaps exist.

Input to the simulation includes network geometry data, intra-link desired or target speed, mean discharge rate for intersection approaches, mean lost time by intersection approach, intersection type, signal controls, entry link flow rates, traffic composition, detector location and type, and short and long term events.

1.4.1 Limitations of NETSIM, Version 4.2

(a) *Surveillance Detectors*

NETSIM allows placement of detectors for actuated controllers. But these detectors do not report any traffic parameters such as volume, occupancy or speed.

(b) *Long Term Event Simulation*

NETSIM can simulate lane blockages as long term events if one or more internal links on a network experiences long term blockages. Blockages due to stalled vehicles or illegal

parking may also be simulated. Long term events can be specified for any channelized or unchannelized lane, but not for a pocket lane. The simulation logic permits only one event at any moment and a blockage simulated on a lane is treated as one event. Therefore, blockages over more than one lane may not be simulated concurrently, as they are treated as separate incidents.

To simulate a long term event, the link number, start time of the event, duration of the event and lane blocked by event must be specified. But the exact location of the event cannot be user-specified because it is stochastically assigned assuming a uniform distribution from the tail of the queue at the downstream node to 40 feet upstream from the upstream node.

1.4.2 NETSIM Enhancements

As part of this research, routines were developed and added to NETSIM to allow a user to specify surveillance detector locations and to output traffic parameters over a specified period. Long term event routines were also modified to allow simulation of multiple blockages of the same duration on any internal link of the network. This permitted the exact location of the blockage to be specified and simulated.

1.4.3 NETSIM Representation of Study Networks and Its Calibration

The study networks were coded for NETSIM based on the data collected from the Los Angeles and Anaheim TMC. The network streets were coded as unidirectional links and the intersections as nodes. The geometry that was represented in the simulation included link lengths, turn pockets, and the number and width of lanes. For the actuated intersections, the dual ring control timing plan data included maximum cycle lengths, offsets, force-offs for all phases, permissible periods, minimum and maximum green times, vehicle extension, yellow change intervals, etc. based on the timing plan database of the TMC. The actuated control detectors and the surveillance or system detector configurations were also represented.

shows a schematic of the NETSIM representation of the Anaheim network.

Network Coding

The network geometry for the networks described was coded for NETSIM as an input file according to the data collected from the traffic management center. The network streets were coded as one-directional links and intersections as nodes. The geometry data included link lengths and turn pockets, number and width of through, left, and right turn lanes from strip maps. Since the surface streets were on flat terrain, a 0% grade was assumed. Fifteen-minute detector history for the network were used as the entry volumes

to external links defined. From the TMC the latest turn counts were used as an estimate of the turn percentages for each intersections. Since no data were available on the vehicle mix, 100% autos were assumed to be on the streets. For those fully actuated and semi-actuated intersections, the dual ring timing plan data were coded. The main streets Katella and Washington for the Anaheim and LA network respectively, were coded as sync phases. The detectors in place were coded as four feet detectors placed at the stop line, on each through lane as calling and count detectors. The system detectors were initially coded as surveillance detectors six feet in length, and according to maps.

No bus operations were included in the coding for either networks. The blockages were simulated as one, two or three lane using 'long term events'. Besides the duration of the blockage, based on the modifications made to NETSIM, the exact location of the blockages were also specified.

1.4.4 Calibration of NETSIM

Originally, when NETSIM was developed it was calibrated for a grid network located in downtown Washington, DC. It was calibrated for data aggregated over 4 minutes (3 cycles) for measures of effectiveness such as average vehicles per hour past the stop line, average vehicles per link, vehicle-minutes, vehicle-miles, total delay, average travel time per vehicle, and average vehicle speed.

Previously, NETSIM had been calibrated on aggregate data such as MOE's for 3 or more cycles for link specific averages over the simulation period for an entire network, or for performance over the entire simulation period. This effort involved using real cyclic data from existing loops for direct field-to-model comparisons to the new surveillance loops for volume and occupancy.

Data set#1 described in the previous section was used for calibration and Data set#2 was used for validation. From the DAN reports the data were averaged over loops on the same approach for every cycle over the hour of both AM and PM peak period. The 15-minute DHS reports were used to compute the half-hour average volumes on the entry links. Once all the geometric inputs described previously were input, including the volume, turn percentages and signal control data, five different seeds were used to run the simulation for an hour. In an effort to better match the simulated data and the field data, the free flow speed, effective vehicle length, effective loop length, queue discharge characteristics, and start-up lost time were modified. In order to calibrate these variables to produce the best fit parameters, each time all but one of the parameters were varied while the rest were held constant.

1.5 Simulated Data Collected

Once the selected networks were represented in NETSIM and calibrated, system loops simulated were used to collect data for these networks. Data for incident-free conditions (i.e. without any operational problems), and for incident conditions were collected. The traffic data that was collected included cyclic volume and occupancy.

1.5.1 Simulated Data Set

Anaheim network:

Thirty-six half-hour morning peak periods were simulated to collect incident-free detector data, where the random seeds of the simulation were varied. The entry volumes to the network were varied based on the 15-minute detector data collected for the network for the morning peak period.

Incidents were simulated on link 241-230, Katella Westbound between Clementine and Harbor. One lane, two lane and three lane blockages were simulated for durations varying from 2 to 16 minutes, with start times randomly varying with respect to cycle length. The locations of the blockages were also varied randomly along the length of the link. In each case the morning (AM) peak period was simulated for approximately 30 minutes (i.e. 14 cycles), where the cycle length was 126 seconds. Data for 108 blockages were collected from the system loops. There were 36 cases each of 1 lane, 2 lane and 3 lane blockages; and for each case durations of 2, 4, 6, 8, 12 and 16 minutes were used. For all 108 cases, the location and start time of the blockage was randomly varied.

Los Angeles Network

For this network, incident-free data were similarly collected from the simulation for both evening and mid-day peak period.

Mid-day periods were simulated for an hour and a half each for thirty days. In this case, only one-lane blocking incidents were simulated for these periods. The duration of the incidents were varied from 7 to 25 minutes, and the location was randomly varied along the length of the link. Incidents were simulated on different links of the network. The entry flows to the network were based on the 15-minute detector data history for the entry links. An evening peak period (PM peak) period was also simulated. Incidents were simulated on different links, with durations varying from 10 to 25 minutes, and locations of one lane blockages varying randomly along the length of the link. Only one-lane blockages were considered as it was expected that these would form the lower limit of detection rates of an incident detection module. A total of 270 incidents were simulated, 90 on each link, with the flow levels varied based on the average of 15-minute volumes (ranging from 700-1100 vehicles per hour) based on incident-free field data collected over several days during the PM peak period. Forty-five minute periods were simulated, with incidents starting 6 minutes after the start of the simulation. For the 45-minutes of simulation, volume and occupancy data were collected every cycle (90 seconds in this case). The hour of simulation was preceded by a 15 minute warm-up period. Similarly, 90 45-minute incident-free periods were also simulated. A total of 13,320 incident and non-incident vectors were used for training and testing.

1. Model Development

1.1 Introduction

The problem of incident detection is one of pattern recognition, and neural networks have been known to perform well for these problems. Pattern recognition has been described as a process of replacing an opaque mapping between some features of an object and its class (Zadeh, 1977). The recognition for a human takes place through perceptual or cognitive systems. But in automating the process of pattern recognition, the opaque mapping can be implemented by a transparent mapping. There are basically two steps to determining the mapping:

1. determining what input features of an object are pertinent to the process of determining its pattern class
2. determining the mapping between the selected input features and its pattern class

Both (1) and (2) are unknown, i.e. what aspects of the problem are pertinent to recognition and what transformation to perform are unknown. For the case of classifying incident and non-incident data, the traffic parameters that best represent the two classes are not known, and the mapping is also not known.

Neural networks have been known to perform well for pattern recognition problems and have been successfully applied to detecting freeway incidents (Ritchie and Cheu, 1990;

Cheu and Ritchie, 1994). It is specially appropriate when the functional relationships can not be expressed in terms of a closed form solution. It is expected that neural networks, by setting the appropriate connection weights, without any a priori structure imposed, will seek to find the function stated in the universal approximation theorem (Hecht-Neilsen, 1987). As shown by researchers (Hecht-Neilsen, 1987; Gallant and White, 1988), multi-layer neural networks can be used as devices to represent arbitrary continuous functions. In more recent work, (Cybenko, 1989; Funashashi, 1989; Hornik et al., 1989) the universal theorem states:

> *Let* $\varphi(.)$ *be a nonconstant, bounded, and monotone-increasing continuous function. Let* I_p *denote the p-dimensional unit hypercube be a nonconstant, bounded, and monotone-increasing continuous function. Let* I_p *denote the p-dimensional unit hypercube* $[0,1]^p$. *The space of continuous function on* I_p *is denoted by* $C(I_p)$. *Then, given any function* $f \in C(I_p)$ and $\varepsilon > 0$, *there exists an integer M and sets of real constants* α_i, θ_i, *and* w_{ij}, *where I=1, ...M and j=1,....p such that we may define*
>
> $$\left|F(x_1,....,x_p) - f(x_1,....,x_p)\right| < \varepsilon \qquad \textit{Eq.}\ 1\text{-}1$$
>
> *for all* $\left\{x_1,.....,x_p\right\} \in I_p$.

This theorem can be directly applied to Multilayer Feedforward (MLF) neural networks using a sigmoidal nonlinearity, which satisfies the conditions imposed on the function φ(.). Therefore, it represents the output of a MLF neural network with p input units, M

hidden layer units, weights w, and threshold θ. The universal approximation theorem described here provides the mathematical justification for the approximation of an arbitrary continuous function as opposed to exact representation, given a training set of x and a target output f(x). Though the existence of such a function is stated in the theorem, no guidelines are available on how to determine the function. Therefore, for problem-specific applications, this needs to be determined.

In attempting to study the feasibility of applying neural network techniques to identify incident patterns from loop data for arterials, several different architectures were explored and a comprehensive system was finally proposed to address the problem of detecting different types of operational problems. Initially the well known multilayer feedforward neural network architecture was used to select the appropriate input features that best represented the pattern class of incident and non-incident conditions.

1.2 Selection of Features

1.2.1 Input Features

First, a set of input variables was selected based on data normally available at a traffic management center. Also included in the study were some data that may be available based on different detector configurations. These variables were added to investigate the effect these variables may have on the overall performance of the algorithm and to evaluate the benefit or performance improvement gained by collecting additional data.

The input features for one set of system detectors per link, normally located 100-350 ft. upstream of a downstream node (Configuration#1, Figure 1-1(i)) include:

- volume
- occupancy
- difference in occupancy between adjacent lanes
- distance between stop line of downstream intersection of a link and system detectors
- block length

For both upstream and downstream detectors per link (Configuration#2, Figure 1-1(ii)):

- volume and occupancy for downstream and upstream detectors
- difference in occupancy between adjacent lanes
- difference in occupancy between upstream and downstream detectors
- distance between upstream and downstream loops

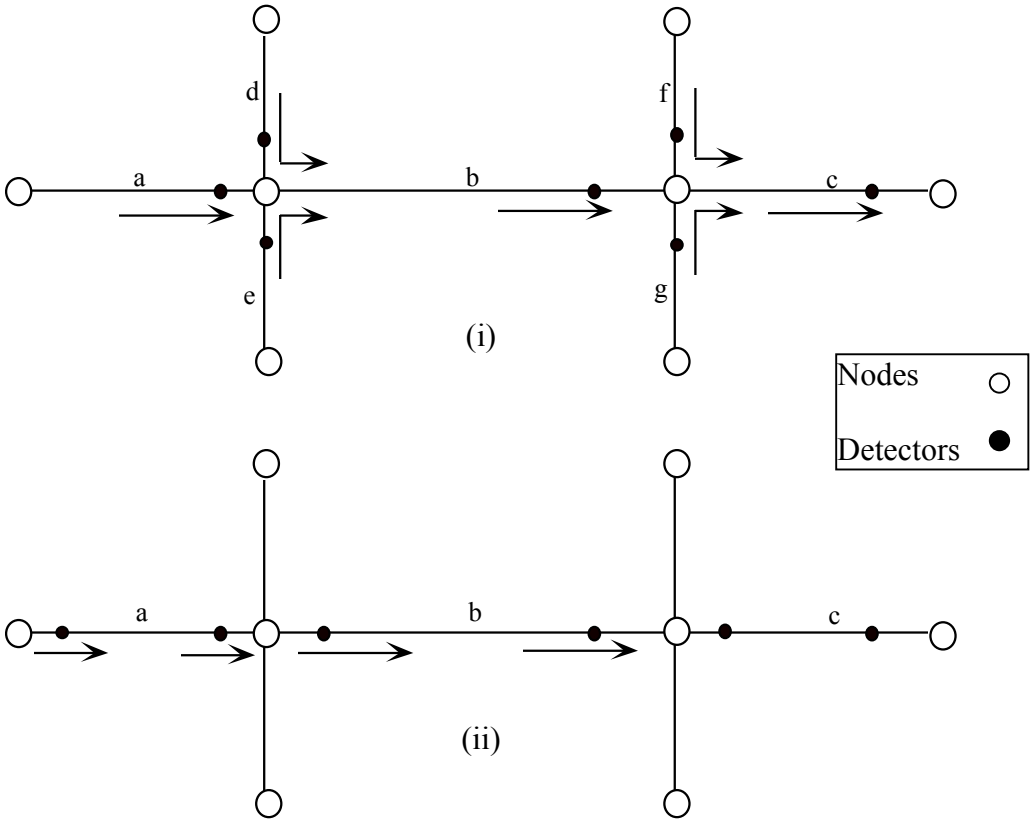

Figure 1-1. Detector (i) Configuration #1 and (ii) Configuration #2

1.2.2 Output Features

The objective of this research is to develop neural network models to detect different types of operational problems for surface street networks. Therefore, the binary outputs of the neural network models define the type of operational problems detected on each link of the street network. The output produced every cycle indicates whether the model detected the presence of a problem on a particular link. When persistence tests are introduced, the incident-output has to persist over the required interval before an incident-alarm is set or the problem is detected by the neural network model.

During training of the MLF neural network models developed, each input vector had to be labeled with an output vector as an incident-vector or a non-incident vector. For simulated data, the labeling of the data set was based on the start time of the simulated incident. For field data, when the start-time of the operational problem was recorded, it was used. But for problems such as special event conditions, the detector data were examined to determine start times. In a few cases, video cameras recorded special event conditions with a time stamp in each case, and these recordings were later analyzed to determine the approximate time these problems occurred.

1.3 Performance Measures

Different performance measures were used during training and testing of the neural network models. They are presented below. Besides these, other statistical techniques used will be discussed in detail.

1.3.1 Root Mean Square (RMS) Error

The square root of the average squared difference between the desired output and the output produced by the neural network model. It is normally used during training to compare alternative models developed; a smaller RMS indicates improvement in the performance of a neural network developed, but it in no way implies good generalization capability, and therefore was also tested on out-of-sample data or a separate test data set to estimate the actual performance of the model.

1.3.2 Detection Rate (DR)

The proportion of total incidents detected expressed as a percentage. For every cycle, the incident detection model was applied, and an incident or incident-free state was identified. But depending on the number of persistence intervals applied, an incident alarm was set. Detection rate was computed based on the total number of operational problems in the data set.

1.3.3 False Alarm Rate (FAR)

Based on an incident-free data set, the proportion of input vectors that were incorrectly classified as incident vectors were identified as false alarms. Out of the total number of applications of the model to incident-free data set, FAR was computed to determine how many incident alarms were falsely set. The FAR's were also computed with the same number of persistence tests as the detection rates (DR).

1.3.4 Average Time To Detection (TTD)

TTD was computed as the average time elapsed between the time the incident started and was detected. Since the reporting interval was cyclic, TTD was expressed in cycles. This interval could vary depending on the base cycle length in place.

1.3.5 Classification Rate (CR)

Of the total number of applications of cycle length data or input feature vectors, the percent correctly classified vectors (includes both incident and non-incident vectors) by the model was computed. This classification rate was used to monitor the progress of neural network training and testing and also to evaluate alternative topologies.

1.3.6 Statistical Techniques Used to Evaluate the Performance of Models Developed

Traditionally in statistical analysis, in evaluating the overall performance of a pattern recognition technique, several methods are used to reduce biases in reporting performance measures, (Sprint, 1989). A simple example is the case when the training

data set used to develop a model is used to test its performance, producing error rates that are biased or are over optimistic. One way to correct this is to use two independent samples of data, a training set and a test set. Other techniques include cross-validation, leave-one-out procedure, jackknifing or bootstrapping (Sprint, 1989). These are nonparametric techniques of reducing biases. Several of these techniques can be used to evaluate the accuracy of any classification rule, depending on the size of the data set available. These techniques have been considered for the evaluation of the neural network models developed not only to reduce bias but also to address problems of very limited field data for certain types of operational problems. It is expected that these methods of analyzing the performance measures will produce closer to true estimates of their performance, instead of reporting over optimistic results.

1.3.6.1 Training and Testing

In this procedure, the performance of a model was evaluated based on a separate data set which was not used during the development of the model - that is classifying on new observations or applications not seen before. This is done by using the classifier on an independent set, where the classification is known, but not used by the classifier. The predicted and true classifications on the data set give an unbiased estimate of the error rate of the classifier. To carry out this procedure, a certain proportion of the data set is randomly selected and used as a testing data set. This procedure can easily be used when the sample size is large, and a test set can be set aside.

1.3.6.2 Cross-Validation

For applications where the sample size is relatively small, cross-validation can be used. Normally, for this procedure, the data set is split into m sub-samples. The classification rate reported is predicted by the performance on (m-1) sub-samples, where 1 sub-sample is used to develop the classifier. The final rate is based on the performance on m sub-samples. In this way, the performance evaluated is estimated efficiently and in an unbiased manner. One difficulty in applying this method is that it is computer-intensive and requires m number of training cycles.

1.3.6.3 Jackknife

In this method, the full estimate of the error of the classifier is estimated, and n further estimates are made omitting one sample or application each time, for a total of n+1 estimates. If the estimate omitting observation i is given by $\hat{\theta}_{(i)}$, and the mean of the $\hat{\theta}_{(i)}$ by $\hat{\theta}_{(.)}$; that is $\hat{\theta}_{(.)} = \sum_i \hat{\theta}_{(i)} / n$, then the jackknife estimate is:

$$\tilde{\theta} = n\hat{\theta} - (n-1)\hat{\theta}_{(.)} \qquad \textit{Eq. 1-2}$$

1.3.6.4 Bootstrap

Another method of removing bias from reporting the performance or error rates in classifiers is to use a method known as bootstrapping. This method is normally used when the sample size is very small. In this method, the classification process is replicated a number of times. A large number of bootstrap replicate samples are created,

each sample being a randomly chosen replicate of the original sample, that is a random sample size of n is taken from the original sample with replacement. Each bootstrap sample is used to train or develop a classifier, which is then used to predict the classes of the original data that were unused in the training set. This gives one estimate of the error for each bootstrap sample, and the average error rates over all the bootstrap samples are then combined to give an estimated error for the classifier or model developed.

It has been shown that mean square error can be broken down into two components, bias and variance; reducing one increases the other. It has been reported (Sprint, 1989) that cross-validation produces estimates that are too scattered, so that the confidence intervals are too wide. The bootstrap procedure gives a narrower confidence interval, but the price to pay is that the estimated error rates are optimistic or biased. The trade-off between bias and random error means that, as a general rule, the bootstrap method is preferred when the sample size is small, and cross validation when the sample size is large. It may be mentioned here that the amount of bias remains the same, when the same method is used for all classifiers. Since bootstrap is the best method to reduce variability, the most effective way to compare classifiers for small data sets is to use bootstrap. For large data sets, cross-validation and bootstrap figures converge. Therefore, for field data sets, where the incident data set was small, the bootstrap method was applied to report the performance of the classifiers developed.

1.4 Neural Network Model Developed

1.4.1 Feasibility Study

To demonstrate the potential of using artificial neural networks models to automate the detection of traffic operational problems on arterial streets in signal controlled networks and for appropriate input feature selection, a section of the Anaheim, California's surface street network was coded for a micro-simulation using NETSIM, as described in Chapter 4.

For the AM peak period, simulated cyclic volume and occupancy data and some geometric data for actual system detectors were collected from the surveillance detectors simulated. Experiments were designed to simulate randomly located lane blockages of varying duration, and detector data from the actual locations of system loops in the network were used to develop a mullti-layer feed forward artificial neural network to detect lane blockages.

1.5 Different Input Features

The study using simulated data for the Anaheim network described in the previous section was used to carry out an investigation to select the appropriate input features for detecting the various types of operational problems discussed in Chapter 1. The intent was also to investigate the development of a basic input feature structure that could be used for the development of a general neural network model.

The basic structure proposed separates the detection of problems on individual links. This attempt identifies a basic unit of setting up a detection module which can be used for other sections of the network to detect problems on other links.

Links were simulated with different types of detector configuration: (a) one set of detectors per link at the downstream end of links as shown in Figure 1-1(i); (b) two sets of detectors, one set close to the downstream end and another close to the upstream end, as shown in Figure 1-1(ii). The intent was to develop a neural network to detect operational problems on link b, using data from through lane detectors on link a, b, c, and turn pocket lane detectors on link d, e, f, g, for detector configuration Set#1. Since normally only through lanes are detectorized, only through lane detectors from link a, b, and c were used. For detector configuration Set#2, data from link b were used only.

A basic detection structure consisted of 78 inputs (for 6 detectors, 12 traffic parameter inputs each, occupancy difference and 2 geometric data) for detector configuration Set#1 (only using detectors close to the downstream end of link). A basic structure was also developed for detector configuration Set#2 (upstream and downstream detectors) which uses (2 detectors per lane, 12 traffic parameters, occupancy difference between upstream and downstream and adjacent lanes, 2 geometric) using 67 input features. Various combinations of input feature were selected and the hidden layer processing elements were varied, along with other backpropagation parameters such as learning rates, and momentum coefficients and a learning schedule was developed.

A series of MLF neural networks were developed to identify the best set of traffic and geometric parameters to include. The input features considered in each case are presented in

Table 1-1. and the procedure followed to develop the neural network model is described below.

1.5.1 Parameter Selection for Multilayer Feedforward (MLF) Neural Network

The parameters that are variable for a MLF network are mainly the number of hidden layer processing elements, the learning coefficient η, and the momentum coefficient α. The number of hidden layer processing element can vary from 2 to ∞, the learning and momentum coefficients can vary from 0 to 1.

1.5.2 Number of Hidden Layer Processing Elements

For pattern recognition problems, the objective of developing the neural classifier is to correctly classify an input feature vector, and for this research it is to correctly classify an input vector on a cyclic basis as either incident or incident-free. Therefore, the selection of number of hidden layer processing elements was based on correct classification rate. Initially an arbitrary large number of processing elements were used with a set of learning

Table 1-1. Input Features

Input Features	*Detector Config#1*					*Detector Config#2*				
	Feature Set#									
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Volume at t-5	x	x				x	x			x
Volume at t-4	x	x				x	x			x
Volume at t-3	x	x	x		x	x	x	x		x
Volume at t-2	x	x	x		x	x	x	x		x
Volume at t-1	x	x	x		x	x	x	x		x
Volume at t	x	x	x		x	x	x	x		x
Occupancy (Occ.) at t-5	x	x		x		x	x		x	x
Occ. at t-4	x	x		x		x	x		x	x
Occ. at t-3	x	x	x	x	x	x	x	x	x	x
Occ. at t-2	x	x	x	x	x	x	x	x	x	x
Occ. at t-1	x	x	x	x	x	x	x	x	x	x
Occ. at t	x	x	x	x	x	x	x	x	x	x
Occ. difference (diff.) between upstream(u/s) and downstream (d/s) detectors at t-3						x	x	x	x	x
Occ.difference between u/s and d/s detectors at t-2						x	x	x	x	x
Occ. difference between u/s and d/s detectors at t-1						x	x	x	x	x
Occ. difference between u/s and d/s detectors at t						x	x	x	x	x
Occupancy difference between lanes at t-3	x	x	x	x		x	x	x	x	
Occupancy difference between lanes at t-2	x	x	x	x		x	x	x	x	
Occupancy difference between lanes at t-1	x	x	x	x		x	x	x	x	
Occupancy difference between lanes at t	x	x	x	x		x	x	x	x	
Distance between d/s detector and stop line	x		x	x	x	x		x	x	x
Block length	x		x	x	x	x		x	x	x
Distance between u/s and d/s detectors						x		x	x	x

rate and momentum coefficients. For each training session, η and α were held constant. The mean-squared error was computed and recorded. Since a closed form solution of the decision boundary can not be determined, an experimental testing of the trained MLF against an independent set of input-output pairs was conducted from a randomly drawn set of input vectors. Therefore, the classification rate, computed as follows provided an estimate of the classification performance:

$$p = \frac{Correctly\ classified\ vectors}{Total\ number\ of\ \ vectors}$$

which calculates the average probability of correct classification estimated for given test set size. Even when the RMS was lowered, if there was no significant improvement in the average probability of correct classification, then the performance of the neural network was considered to have not improved.

1.5.3 Optimum Learning and Momentum Coefficients

There are several criteria that can be applied to the selection of the optimal learning and momentum coefficients. They can be based on attaining the lowest error or the best generalization in the least number of epochs. For this study, the lowest error over the least epochs was selected and therefore the ratio of RMS and epoch was plotted for various combinations of the coefficients to select the best parameters.

1.5.4 Generalization

To ensure good generalization ability of the neural network model developed, for a fixed network architecture, the best training set size was determined. In the neural network literature, the question of finding the adequacy of the training set size has attracted a lot of attention. Baum and Haussler (1989) present a case of a network with a single hidden layer for a binary classification problem that suggests the following criteria:

$$N \geq \frac{32W}{\varepsilon} \ln\left(\frac{32M}{\varepsilon}\right)$$

This is approximated as:

$$N > \frac{W}{\varepsilon}$$

where, M is the total number of hidden layer processing units, N denotes the total number of random samples used during training, W represents the total number of weights, and ε denotes the fraction of errors permitted. This was used as a guideline for determining the size of the training pattern set.

1.6 Output Features

A neural network model to develop three types of operational problems - lane blocking incidents, special events and detector malfunctioning were developed. A single neural network model to detect one or more types of problems were developed as shown in Figure 5-2. For the first set of models developed ((a)), both for the multilayerfeedforward neural network (MLF) and the projection network (PN), a single

binary output was trained and tested as non-incident state (output:0) and incident state (ouput:1) for one or more types of operational problems. For the second set of single neural network models developed, the number of input features used were based on the types of operational problems being tested (Figure 5-2(b)). For example, to develop a single MLF or PN network for three different types of operational problems, three binary outputs were used.

1.7 Modular Network Developed

As described in the previous chapter, when tasks are assigned to different models addressing different aspects of a problem, more efficient and suitable methods of solving specific problems can be developed. In the case of detecting different types of traffic operational problems, a modular architecture was used as shown in Figure 1-2.

To demonstrate the potential of using a modular architecture as presented in an earlier section, simulated field data collected for the lane blocking incidents, special event condition and detector malfunction described were used.. The gating mechanism was able to assign separate tasks to separate networks. This network structure was also extended to address detecting problems during different operating conditions.

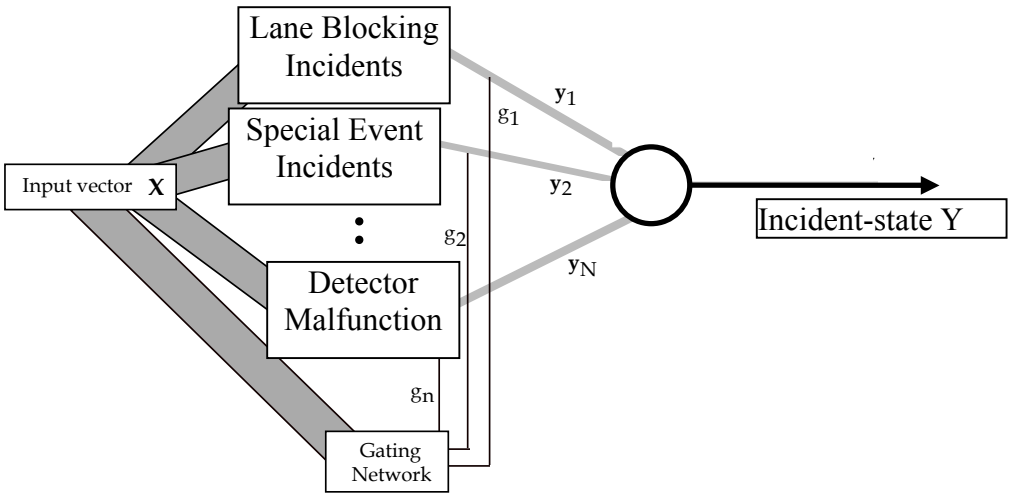

Figure 1-2. Modular Architecture of Neural Network Models to Detect Different Types of Operational Problems

1.7.1.1 Different Types of Operational Problems

As discussed in Chapter 1, the detection of operational problems should include detection of several different types of problems. In this research, lane blocking incidents, incidents due to special events, and detector malfunctioning were addressed.

1.7.1.2 Training and Testing of Modular Network

Modular networks were trained using data for incident-free, lane-blocking incidents, special event-detector malfunctioning for the Los Angeles and Anaheim network described in Chapter 4.

1.8 Statistical Classifiers Developed

1.8.1 Discriminant Analysis

As described in Chapter 3, discriminant analysis, a pattern recognition technique has been proposed for detecting incidents (Chen and Chang, 1993). For the field and simulated data collected for the test networks, a discriminant analysis was performed to evaluate its performance. The same data set that was developed to train and test the

neural network models were used to similarly develop the discriminant classifier and evaluate its performance. The best input feature set selected were used to develop the classifier. All the variables or input features used in the neural network development (feature set number 3 and 8) were found to be significant. Also included were the additional variables that were used in the Chen and Chang (1993) study. But the additional variables such as distance between the detectors and the block length were also included for further analysis and testing of significance. The significance of the overall model was also considered in the evaluation. The results of the classifiers performance are presented in the next chapter. In order to develop the Bayesian classifier to classify an incident or incident-free state from the discriminant score, the prior probabilities of the groups were considered to be equal.

In the analysis, first the discriminant scores were determined by the discriminant function developed. The discriminant function was developed by a linear combination of the input features, where the coefficients were selected so that the values of the discriminant function differ as much as possible between the incident and non-incident groups. The discriminant score was then used to develop a classification rule, where the probabilities of belonging to a particular group was determined. The group with the highest probability was the group the input vector was assigned to.

1. Results and Comparative Evaluation

1.1 Introduction

This chapter reports the results of the different types of neural classifiers investigated and compares their performance with the statistical classifiers described in Chapter 3, for data collected from a micro-simulator for the study networks selected, and also for limited field data collected. Performance measures such as detection rate (DR), false alarm rate (FAR) and average time to detection (TTD) were computed to compare the performance of the different neural networks and statistical classifiers. But to compare neural networks of the same architecture, the root mean square error (RMS) and classification rates (CR) were used. The multi-layer feedforward neural network was used to select the appropriate input features, and these features were used for all subsequent analyses. Single multi-layer feedforward and projection networks were developed to detect different types of operational problems, and a modular network was also developed to compare the performance of the modular and the equivalent single network. The best results of these networks were then compared with the results obtained from using a statistical classifier, for the same data set.

1.2 Neural Network Classifier

Two types of neural network classifiers, the multilayer feedforward neural network and the projection network, were investigated. For these networks, the different input feature sets described in the previous chapter 5 were used, and a single output feature to determine incident (output state: 1) and non-incident (output state: 0) was used. For the single network developed to detect the different types of operational problems, the same output structure was used. A modular architecture was also developed using the input features selected, but different modules detected the different types of operational problems. The following section presents the results of the input feature and the neural network parameter selection process.

1.2.1 Multilayer Feedforward Neural Network (MLF)

1.2.1.1 Feasibility Study Results

Table 1-1 presents the results for the test data set. For all the test cases a persistence test (PT) of one interval (when the incident state remained for two consecutive intervals) was used. Including a persistence reduced the false alarm rate and decreased the detection rate. Overall, the FAR was 1.16% for 0 PT and 0.23% for 1 PT, and the TTD was 1.63 cycles without any persistence test.

As expected, and as reported in the freeway incident detection case, there is a trade off to be made when false alarm rates are reduced and the detection rates are increased.

Table 1-1. DR and TTD Performance Measures

Incident Duration (min.)	*1 lane blocked*		*2 lane blocked*		*3 lane blocked*	
	0 PT	*1 PT*	*0 PT*	*1 PT*	*0 PT*	*1 PT*
2	0	0	33	0	67	0
4	67	50	67	50	100	67
6	67	67	67	67	100	100
8	67	67	100	67	100	100
12	67	67	100	67	100	100
16	67	67	100	67	100	100
Overall DR	55	53	78	50	94.5	78

The initial analysis performed here used only volume and occupancy data. The results obtained for the Anaheim network were very encouraging, and further work was continued to include other input features and test neural networks developed for a variety of network and detector configurations for simulated and field data. Based on this initial study, issues of transferability, and detection of other types of operational problems not included in the initial study were addressed later as described in the next sections.

1.2.1.2 Input Feature Selection Results

Table 5.1 described the various input features considered for each particular detector configuration. For a specific input feature set considered, an optimum set of neural network parameters was selected for each architecture. For example, for the MLF using input feature set#1, the best set of learning and momentum coefficients and number of hidden layer processing elements was selected as described in the previous chapter. The results presented here are for the best set of parameters selected.

The best input feature set selected for detector configuration #1 was number 3, and for detector configuration #2 was number 8, based on the performance measures - DR and FAR. The results of only a subset of the feature sets presented in Figure 1-1 are shown, since the performance of the others was considered unsatisfactory after preliminary testing. The results presented include feature set#3, 4, 5 and set# 8, 9, 10 for detector configuration #1 and 2 respectively. This evaluation was performed to select the input features to be used for further development of incident detection models. As shown by the figure, models developed using Set#2 detectors outperformed those using Set#1 detectors. The results in terms of performance measures DR and FAR are presented for 0, 1, 2 and 3 intervals of persistence test. It may be mentioned here that with the introduction of persistence tests, an incident state had to persist for additional consecutive intervals equal to the persistence, before an incident alarm could be declared or an incident could be said to have been detected. For example, with a 3 interval persistence test, if the outputs over 6 intervals was 0,1,1,0,1,0 (with "1" indicating an incident state)

then an incident alarm was not declared or the incident was not considered detected as the incident state did not persist for 3 consecutive intervals. This is an important distinction to make

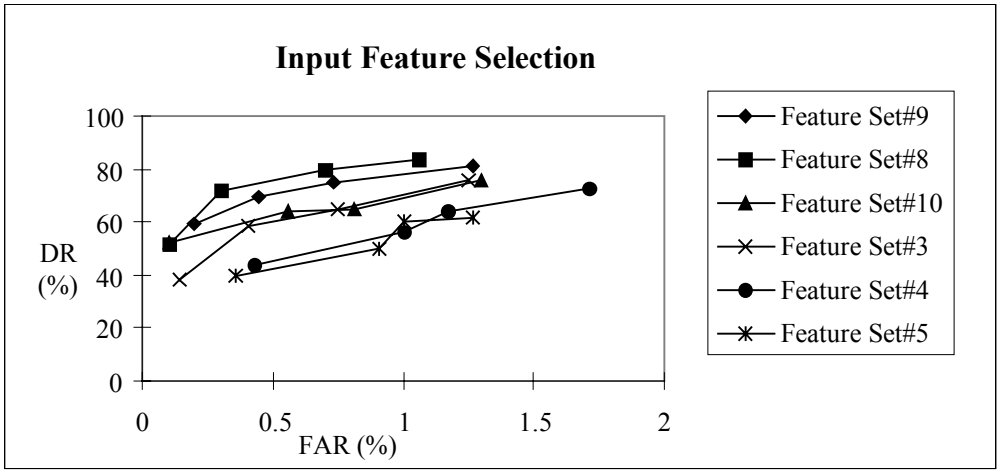

Figure 1-1. Input Feature Selection Using Simulated Data

to avoid over estimating the DR or the overall performance of the model. Initially, two or more lane blockages were included, but were later dropped since the performance of one-lane blocking incidents could provide the more conservative estimate of the model. The results presented here were obtained by using simulated data for the Anaheim network for lane blocking incidents.

Based on Figure 1-1, for detector configuration Set#1, volume, occupancy, and occupancy difference between adjacent lanes for t, t-1, t-2, t-3 cycles were selected as input features. For detector configuration Set#2, volume, occupancy and difference in occupancy between upstream and downstream detectors, and difference in occupancy between adjacent lanes for t-3, t-2, t-1, t cycles were selected as input features.

Next, for the same set of input features selected, special event condition cases were included to test the performance of the models. As shown in Figure 1-2, the performance of the MLF deteriorated. Since in this case where a single MLF was developed to detect the different types of operational problems, and the output vector for both types was labeled as an incident vector - it was not possible to separate the results based on type of problem detected. Detector malfunction could not be included in this evaluation, because the simulation could not produce detector data for this type of operational problem. This type was included later when field data were used in the analysis.

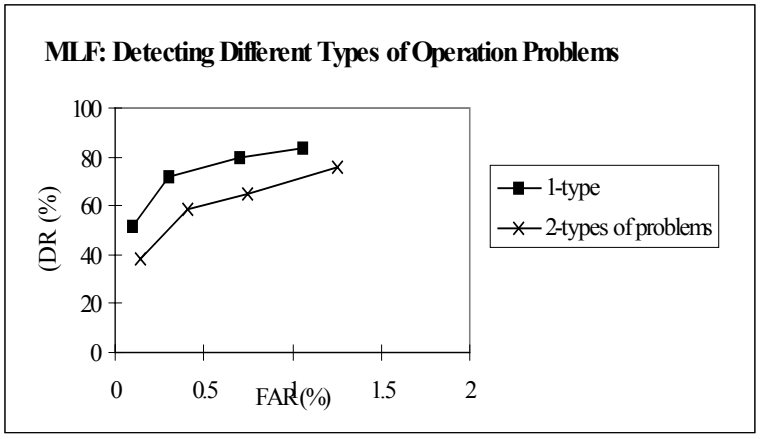

Figure 1-2. Single MLF Network to Detect Different Types of Problems

1.2.2 Projection Network

For input feature set# 3 and 8, and simulated data for the Los Angeles network described previously, a projection network was developed. The results from training and testing on different data sets did not show any improvement in performance over the multi-layer feed-forward network. The only difference between the MLF and the projection network was observed during training. As shown in Figure 1-3, the performance of the PN was not significantly better than that of MLF, but the number of training cycles required to train the network was significantly less that that of the MLF.

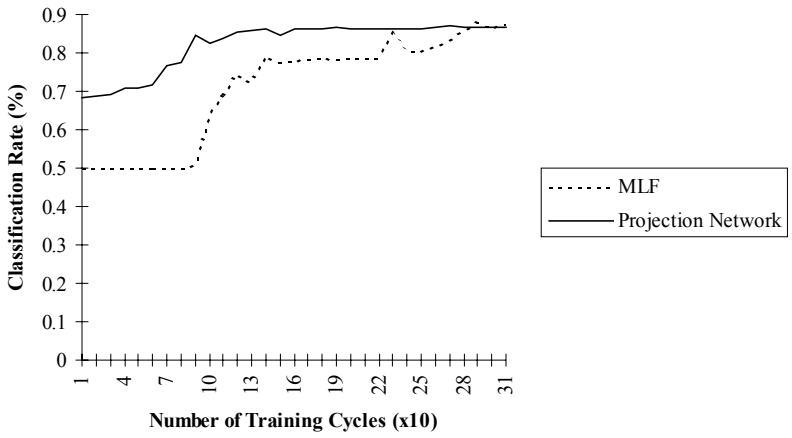

Figure 1-3. Training of MLF and Projection Network

1.2.3 Modular Neural Network

Initially, the MLF was developed to detect different types of operational problems using the same network. But as described in the previous chapter, a modular network was developed to assign the detection of different types of operational problems to different modules. A gating mechanism determined the assignment to different modules. The results for both the single MLF network and the modular network are shown in Figure 1-4 for field data. Both field and simulated data results show that the performance of the model improves for the modular architecture.

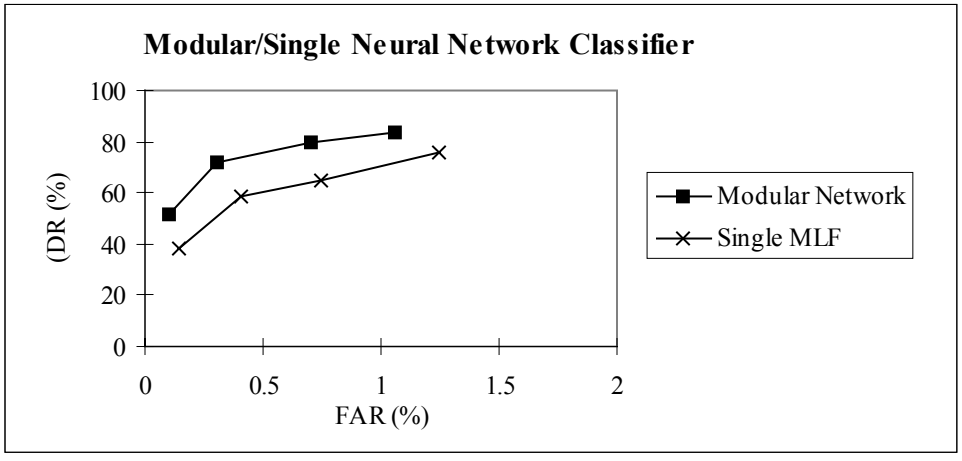

Figure 1-4. Single and Modular Neural Network Model

1.3 Neural Network and Statistical Classifiers

Multilayer feedforward neural networks were developed using simulated data for the Los Angeles network to compare their performance with a statistical classification method - discriminant analysis. The comparison was made for not only simulated data but also for field data. The results were also consistent for both cases - the MLF outperformed the discriminant classifier for the same input feature set and also for the same data set, and persistence tests. Figure 1-5 presents these results. In this case, only the DR and the FAR were used as measures for comparison. As the figure illustrates, the MLF performed better for every point on the performance envelope.

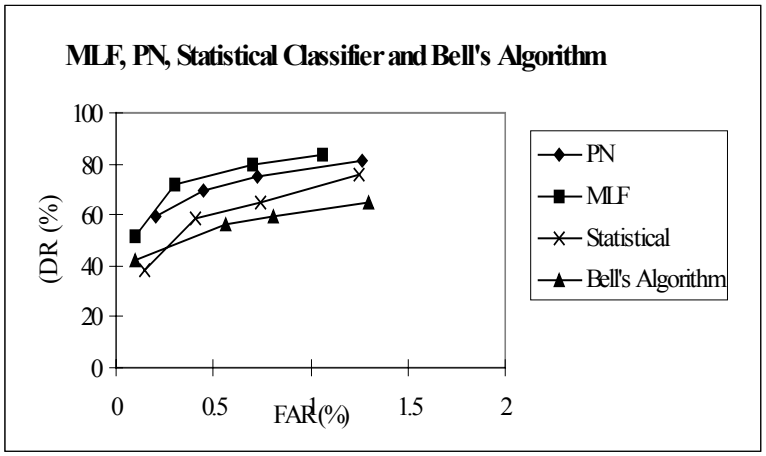

Figure 1-5. Performance of Neural Network and Statistical Models

1.4 Effect of Flow Conditions and Network Geometry on Performance

When the neural network developed was tested for different flow levels (high and medium flow during mid-day and evening peak periods), performance varied. The same tests were performed for different block lengths, which revealed some interesting results. Overall, the detection algorithm performed better during higher flows, perhaps as the vehicles are not able to maneuver around an incident during high flows, queues form, and occupancies increase and therefore are more readily detected. The results for different links reveal that not only does the flow level have an affect, but also the block length has a more pronounced effect on the performance on the model for the same detector configuration (Figure 1-6). This is perhaps because for larger block lengths and during lower flows, vehicles are able to change lanes and maneuver around incidents more easily, and therefore the patterns of incidents are more difficult to isolate from non-incident patterns of traffic.

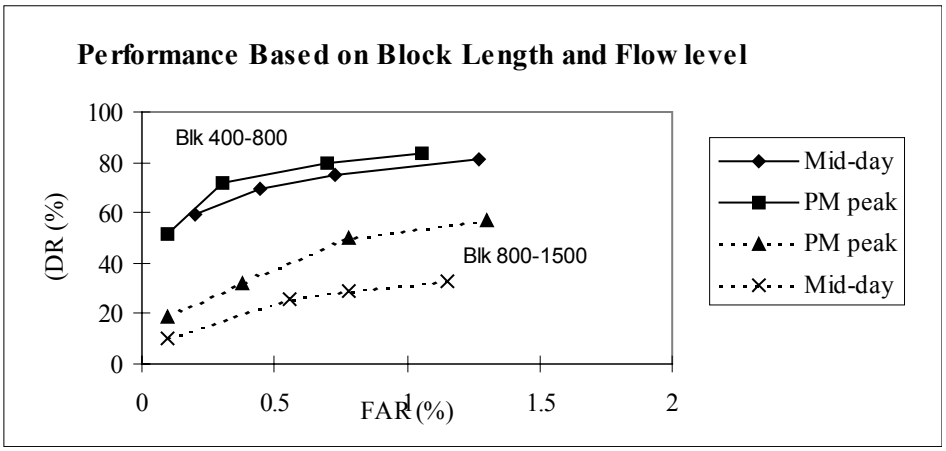

Figure 1-6.Performance of the Modular Neural Network Model Based on Block Length and Flow Levels

1. References

Ahmed, S.A. and Cook, A.R., (1982), "Application of Time-Series Analysis Techniques to Freeway Incident Detection", *Transportation Research Record*, Vol. 841, pp. 19-21.

Bell, M.G.H. and Thancanamootoo, B. (1988). Automatic Incident Detection within urban traffic control systems. Proceedings of the Roads and Traffic 2000 Conference Vol. 4(2). Berlin.

Bretherton, R.D. Bowen, G.T. (1991). Incident Detection and Traffic Monitoring in Urban Areas, Proceedings of DRIVE Conference on Advanced Telematics in Road Transport, Vol. 1.

Chen, C. and Chang, G. (1993). "A Dynamic Real-Time Incident Detection System for Urban Arterials - System Architecture and Preliminary Results", Proceedings of the ASCE Third International Conference on Applications of Advanced Technologies in Transportation Engineering, Seattle, Washington.

Cheu, R.L. and Ritchie, S.G. (1994). Neural Network for Automated Detection of Freeway Incidents. International Conference on Advanced Technologies in Transportation and Traffic Management, Singapore.

Duda, R.O., and P.E. Hart, (1973). *Pattern Classification and Scene Analysis*. New York: Wiley.

Elman, J.L. (1990). "Finding structure in time." *Cognitive Science* **14**, 179-211.

Eshraghnia-Jahromi, A. (1992). Arterial Delay and Performance Seen Through Traffic Models, Ph.D. Dissertation.

Federal Highway Administration (1992). TRAF User Reference Guide, FHWA-RD-92-060.

Han, D.L. and May, A.D. (1988). Artificial Intelligence Approaches for Signalized Network Control, Working Paper, University of California, Berkeley.

Han, D.L. and May, A.D. (1989). Automatic Detection of Traffic Operational Problems on Urban Arterials, Research Report, University of California, Berkeley.

Han, D. L. and May, A.D.,(1990). "Traffic Flow Characteristics of Signalized Arterials under Disturbance Situations", *Research Report*, University of California, Berkeley.

Heicht-Nielsen, R. (1990). Neurocomputing, Addison-Wesley Publishing Company.

Hrycej, T. (1990). "A Modular architecture for efficient learning*"." International Joint Conference on Neural Networks*, Vol. 1, pp.557-562, San Diego, California.

Jacobs, R.A. and M.I. Jordan, (1991)."A competitive modular connectionist architecture." In Advances in Neural Information Processing Systems 3 (R.P. Lippmann, J.E. Moody, and D.J. Tourezky, eds), pp.767-773. San Mateo, CA: Morgan Kaufmann.

Jacobs, R.A., M.I. Jordan, S.J. Nowlan, and G.E. Hinton, (1991). "Adaptive mixtures of local experts." *Neural Computation* **3**, 79-87.

Jacobs, R.A., M.I. Jordan, and A.G. Barto, (1991). "Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks." *Cognitive Science* **15**, 219-250.

Kosko, Bart, (1992). Neural Networks and Fuzzy Systems: A Dynamical systems approach to machine Intelligence, Prentice Hall.

Levin, M. and Krause, G.M., (1978). "Incident Detection: A Bayesian Approach", Transportation Research Record", Vol.682.

Michalopolous, P.G., (1991). "Incident Detection Through Video Image Processing, Applications of Advanced Technologies in Transportation Engineering, Proceedings of the Second International Conference.

Minsky, M. and Papert, S. (1969). Perceptrons, MIT Press.

Peat, Marwick, Mitchell & Company (1971). Network Flow Simulation for Urban Traffic Control System - Phase II, FHWA-RD-73-83-87.

Persaud, B. and Hall, F.L. (1989), "Catastrophe Theory and Pattern in 30 Second Freeway Traffic Data-Implication for Incident Detection", Transportation Research Vol. 23A, No.2, pp.103-113.

Rathi, A.J. and Santiago, A. (1990). The New NETSIM Simulation Program, Traffic Engineering and Control, Vol. 31, No. 5.

Rathi, A.J. and Santiago, A.(1990).Urban Network Traffic Simulation: TRAF-NETSIM Program, Journal of Transportation Engineering, Vol.116, No.6.

Ritchie, S.G. and Cheu, R.L. (1993). Simulation of Freeway Incident Detection Using Artificial Neural Networks, *Transportation Research*-C, Vol.1. No.3. pp. 203-217.

Ritchie, S.G., Kaseko M.S., and Bavarian B. (1992) Development of an Intelligent System for Automated Pavement Evaluation. *Transportation Research Record*. 1131, 112-119.

Rosenblatt, F., (1958), "The perceptron: a probabilistic model for information storage and organization of the brain." *Psychological Review* **65**, 386-408.

Rumelhart, D.E. and J.L. McClelland, (1986), "Parallel Distributed processing: explorations in the microstructure of cognition," Vol. 1. Cambridge, MA: MIT Press.

Sellam S., Boulmakoul A., and Pierrelee J.C. A Distributed Real Time Knowledge-Based System Using Video Image Processing for Junctions Automatic Incident Detection, Proceedings of the Second International Conference on Applications of Advanced Technologies in Transportation Engineering, ASCE.

Serano, T.J. (1990). NETSIM: A Computer Program For Simulating Detection and Location Capability of Regional Seismic Networks, San Diego: Science Applications International Corporation.

Sulzberg, J.D. and Demetsky, M. J. (1991). Demonstration of TRAF-NETSIM for Traffic Operations Management, Virginia Transportation Research Council.

Thancanamootoo, S. and Bell, M.G.H. (1988). Automatic Detection of Traffic Incidents on a Signal-Controlled Road Network. Transport Operations Research Group - Research Report No. 76, University of Newcastle upon Tyne.

Tsai, J. and Case, E.R., "Development of Freeway Incident Detection Algorithms by Using Pattern Recognition Techniques", Transportation Research Record, Vol. 722, 1979.

Rourke, A. and Bell M.G.H., (1991)."Traffic Queue Detection Using Image Processing",, Applications of Advanced Technologies in Transportation Engineering, Proceedings of the Second International Conference.

Sellam S., Boulmakoul, A. and Pierrelee, (1991). "A Distributed Real Time Knowledge-Based System Using Video Image Processing for Junctions Automatic Incident Detection", Applications of Advanced Technologies in Transportation Engineering, Proceedings of the Second International Conference.

Thancanamootoo, S and Bell, M.G.H, (1988). "Automatic detection of traffic incidents on a signal-controlled road network", Research report, University of Newcastle upon Tyne, Transport Operations Research Group no. 76.

Werbos, P. (1974)."Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences", Ph.D. Thesis, Harvard University, Cambridge, MA.

Wilensky, G.D. and Manukian, N. (1992)."The Projection Neural Network", International Joint Conference on Neural Networks, June 7-11, Baltimore, MD, Vol. II, pp. 358-367.

Williams, J.C. (1986). Urban Traffic Network Performance: Flow Theory and Simulation Experiments, Ph.D. Dissertation, University of Texas, Austin.

Williams, J.C., Mahmassani, H., and Herman R. (1984). Analysis of Traffic Network Flow Relations and Two Fluid Model Parameter Sensitivity, University of Texas at Austin.