Title
Neural Network Models For Automated Detection Of Non-recurring Congestion

Authors
Ritchie, Stephen G.
Cheu, Ruey L.

Publication Date
1993

CALIFORNIA PATH PROGRAM
INSTITUTE OF TRANSPORTATION STUDIES
UNIVERSITY OF CALIFORNIA, BERKELEY

Neural Network Models for Automated Detection of Non-Recurring Congestion

Stephen G. Ritchie
Ruey L. Cheu

University of California, Irvine

PATH Research Report
UCB-ITS-PRR-93-5

This work was performed as part of the California PATH Program of the University of California, in cooperation with the State of California Business, Transportation, and Housing Agency, Department of Transportation; and the United States Department of Transportation, Federal Highway Administration.

June 1993
ISSN 1055-1425

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

DISCLAIMER

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the State of California or the Federal Highway Administration. This report does not constitute a standard, specification or regulation.

ACKNOWLEDGMENTS

The research reported in this paper was supported by the Partners for Advanced Transit and Highways (PATH) program of the California Department of Transportation under Agreement No. 65H998, Memorandum of Understanding No. 31, and also by the National Science Foundation under grant No. MSM-8657501.

We would also like to thank Dr. Stephen Cohen from the Federal Highway Administration for providing with us the latest working version of the INTRAS model, and Dr. Behnam Bavarian for his contribution in initial discussions on artificial neural networks.

EXECUTIVE SUMMARY

This research project involved the first year of a proposed multi-year research effort to investigate, assess, and develop neural network models from ' the field of artificial intelligence for automated detection of non-recurring congestion in integrated freeway and signalized surface street networks. The basic objective in the first year of the research, the results of which are presented in this report, was to demonstrate the initial feasibility of a neural network approach to freeway incident detection, with the expectation of then developing more advanced techniques for both freeways and signalized surface street networks in subsequent studies. A major source of traffic delay in many large urban areas in the United States is non-recurring congestion caused by incidents such as accidents, disabled vehicles, spilled loads and other special or unusual events, that disrupt the normal flow of traffic. In the last several decades, a number of incident detection algorithms have been developed for freeway surveillance and control systems. A well-known example is the set of algorithms known as the California algorithms which were developed in the 1970's for the California Department of Transportation (Caltrans). However, existing incident detection algorithms have demonstrated varying levels of capability in terms of performance criteria such as detection rate, false alarm rate, and the mean time to detect incidents. The need for high-performance incident detection techniques is particularly great with the advent of intelligent vehicle-highway system WI-IS) concepts for integrated freeway and arterial networks. These systems will rely heavily on an ability to automatically detect non-recurring traffic congestion. In this research project, we hypothesized that spatial and temporal traffic patterns could be recognized and classified by an artificial neural network. This report presents the results of our initial investigation of such models for the automated detection of lane-blocking incidents on an urban freeway. Based on data obtained from a microscopic freeway traffic simulation model, and comparisons with the California incident detection algorithm, the results suggest that neural network models have the potential to achieve significant improvements in incident detection performance.

CHAPTER 1

INTRODUCTION

1.1 THE PROBLEM

A major source of traffic delay in many large urban freeway systems in the United States is non-recurring congestion caused by incidents such as accidents, disabled vehicles, spilled loads, temporary maintenance and construction activities, and other special or unusual events, that disrupt the normal flow of traffic. For example, estimates of the proportion of urban freeway delay in the U.S. attributable to non-recurring congestion range up to about 60% (Lindley, 1987), and this proportion is believed to be increasing. The automated detection of freeway incidents is an important function of a freeway traffic management center, examples of which are increasingly to be found in large urban areas (Judycki and Robinson, 1992). Successful detection of incidents in their early stages is vital for formulating effective response strategies. These may involve real-time control of traffic entering and on the freeway, provision of real-time traveler information, and timely dispatch of emergency services and incident removal crews.

In the last several decades, a number of incident detection algorithms have been developed or proposed for freeway surveillance and control

systems, although few algorithms have apparently been implemented in practice. Examples of the techniques used include decision trees for pattern recognition (Payne et al, 1976), time series analysis (Ahmed and Cook, 1982), Kalman filters (Willsky et al, 1980) and more recently catastrophe theory (Persaud and Hall, 1989), among others. A well-known example in North America is the set of algorithms known as the California-type algorithms (Payne et *al,* 1976) developed in the 1970's for the California Department of Transportation (Caltrans). These algorithms have been used for many years for incident detection in the freeway system in Southern California (Arcenaux et al, 1989). However, existing algorithms have demonstrated varying levels of capability in terms of performance criteria such as detection rate, false alarm rate, and the mean time to detect incidents. The need for high-performance incident detection techniques is particularly great with the advent of intelligent vehicle-highway system (IVHS) concepts for integrated freeway and arterial networks. These systems will rely heavily on the ability to automatically detect non-recurring traffic congestion in order to optimally control both arterial and freeway networks. More effective incident detection methods should result in faster incident response in terms of emergency services and traffic management, and reduced motorist delay and expense due to reductions in non-recurring congestion. One promising approach toward this goal involves the application of neural networks from the field of artificial intelligence.

In recent years, there has been rapid advancement of artificial neural network technology. Neural networks are parallel distributed information

processing architectures that are suited to hardware implementation and real-time operation. They are based on very simplified models of the operation of the human brain. Neural networks consist of many simple processing elements (PE's) that have densely parallel interconnections. A single PE can receive inputs, weighted by the strength of the associated connection weights, from many other PE's, and can rapidly communicate its outputs, if any, to many other PE's. Information flow is thus represented in a parallel and distributed fashion, across the interconnections. Such networks have exhibited learning, memory, an ability to handle "noisy" real-world data, and other significant capabilities.

1.2 RESEARCH OBJECTIVES

One of the principal applications of neural networks has been to pattern recognition problems (Simpson, 1990). We hypothesize that spatial and temporal patterns in traffic data can be recognized and classified by a neural network, and this approach was investigated in this research for detecting urban freeway incidents.

This research addressed the first year of a proposed multi-year research effort that would investigate, assess, and develop neural network models from the field of artificial intelligence for automated detection of non-recurring congestion in integrated freeway and signalized surface street

networks. The basic objective in the first year of the research, the results of which are presented in this report, was to demonstrate the initial feasibility of a neural network approach to freeway incident detection, with the expectation of then developing more advanced techniques for both freeways and signalized surface street networks in subsequent years. Work in both these areas is in fact ongoing and supported by the California Department of Transportation at the University of California, Irvine.

1.3 STRUCTURE OF REPORT

The body of this report consists of an Executive Summary and ten chapters, organized as follows:

Chapter 1 Introduction defines the problem, research objectives and outlines the structure of this report.

Chapter 2 Review of Incident Detection Algorithms discusses different incident detection algorithms for freeways and arterial streets, and their performance measures.

Chapter 3 Review and Selection of Neural Network Models presents an introduction to artificial neural networks and different classifications of

neural network models. This chapter also includes the selection of neural network models for incident detection application.

Chapter 4 The Multilayer Feed Forward Neural Network presents the topology, operations and learning algorithms of the multi-layer feed-forward neural network used in this study.

Chapter 5 Preliminary Case Study presents the results of a feasibility study to demonstrate the use of a multi-layer feed-forward neural network to detect incidents on a freeway section, using artificially generated loop detector data.

Chapter 6 Design of Neural Net Paradigms discusses the disadvantages assocaiated with conventional incident detection algorithms and patterns in detector data, which leads to the design of input and output features of neural network models.

Chapter 7 Training Data contains a description on the use of a microscopic freeway traffic simulation models to generate loop detector data for training and testing of our proposed neural network models.

Chapter 8 Results of Neural Net Training describes the training procedure and the results obtained for the different neural networks.

Chapter 9 Off-Line Incident Detection Performance compares the performance of neural network models in detecting incidents with the California algorithm, using an independently simulated data set.

Chapter 10 Conclusions and Further Research summarizes the results of this study and identifies several future research directions.

CHAPTER 2

REVIEW OF INCIDENT DETECTION DETECTION ALGORITHMS

2.1 INTRODUCTION

During the past several decades, a number of incident detection algorithms have been developed, and to varying extents, tested. In this chapter, we briefly review existing freeway and arterial incident detection algorithms. Particular attention is paid to the two algorithms currently in use: the California algorithm and McMaster algorithm.

2.2 PERFORMANCE CRITERIA

Common measures in evaluating an incident detection algorithm are the detection rate (DR), false alarm rate (FAR) and time-to-detection (TTD). The DR is defined as

$$DR = \frac{\text{no. of incidents detected by the algorithm}}{\text{total no. of incidents}} \times 100\% \qquad [2.1]$$

The FAR has been defined in two slightly different ways in different studies. The first definition is

$$FAR_1 = \frac{\text{no. of detection intervals which give false alarms}}{\text{no. of intervals in the entire data set}} \times 100\% \qquad [2.2]$$

The denominator of FAR, may consist of actual intervals. Figures quoted from on-line tests are usually computed from this definition. The second definition is sometimes called the off-line FAR:

$$FAR_2 = \frac{\text{no. of incident-free intervals which give false alarms}}{\text{total no. of incident free intervals}} \times 100\% \qquad [2.3]$$

The FAR_2 is often called the on-line FAR because it applies to incident-free data during off-line evaluation. The mean TTD for a set of n incidents is given by:

$$\text{mean TTD} = \frac{1}{n}\sum_{i=1}^{n}(t_{ia} - t_{io}) \qquad [2.4]$$

where t_{ia} is the time when alarm of incident i is declared, and t_{io} is the time of occurrence of ith incident. In the event that the occurrence time of an incident is not known, an estimate may be deduced from sources such as loop detector data or police reports.

Both the DR and FAR measure the effectiveness of an algorithm while the mean TTD reflects its efficiency. Unfortunately, the DR and

FAR are positively correlated. This is because to detect more incidents, the thresholds of conventional algorithms have to be more lenient and this inevitably causes some incident-free situations to be included as alarms. The mean TTD is useful in comparing the relative efficiency of two algorithms. It can also be used to compare an algorithm with other non-automated incident detection techniques.

Researchers have not discussed the ranking of the importance of the three evaluation criteria. However, it has been implied that DR should be the most important by fixing it an acceptable value, usually a minimum of 90-95% (Levin and Krause, 1979), and then seeking to reduce the FAR. Placing DR as the most important criterion is logical because the objective of incident detection is to detect as many capacity reducing incidents as possible. The tolerance level of FAR has not been discussed in detail in literature. Only Persaud et al (1990) have noted that the FAR should be less than 0.01% or equivalent to four false alarms per day. A common method used to reduce FAR us to perform a persistence test (i.e. to test for a few consecutive warnings before issuing an alarm). This is based on the fact that many false alarms are caused by random fluctuation of traffic flow. Past results have shown that FAR can be reduce by increasing the duration of a persistence test. However, the inclusion of a persistance test increases the mean TTD, which reduces the efficiency of the algorithm. In summary, the three evaluation criteria are inter-related. The order of importance is typically (1) DR; (2) FAR; and (3) mean TTD.

2.3 FREEWAY ALGORITHMS

2.3.1 California Algorithms

The California algorithms are probably the best known freeway incident detection algorithms. These algorithms are based on the recognition of detector occupancy patterns at sensor station upstream and downstream of an incident location. The structure of the algorithms resemble binary decision trees. The original California algorithm (algorithm #2) is shown in Figure 2.1.

The detection logic is based on the following scenario. When a lane-blockage occurs between two adjacent stations, a vehicle queue will form at upstream of the incident location. The upstream detector station will register high occupancy value while the downstream station will have very low occupancy. Thus, the first two tests in the California algorithm #2 are to compare the absolute occupancy difference (OCCDF) and percentage occupancy difference (OCCRDF) between these two stations against their respective thresholds. In addition, at the onset of an incident, the occupancy of the downstream detector will drop sharply from the normal value. The third test involves the rate of decrease of occupancy value (DOCCTD) at the downstream station. It also distinguishes incidents from recurring congestion. For recurring congestion, DOCCTD will remain approximately constant.

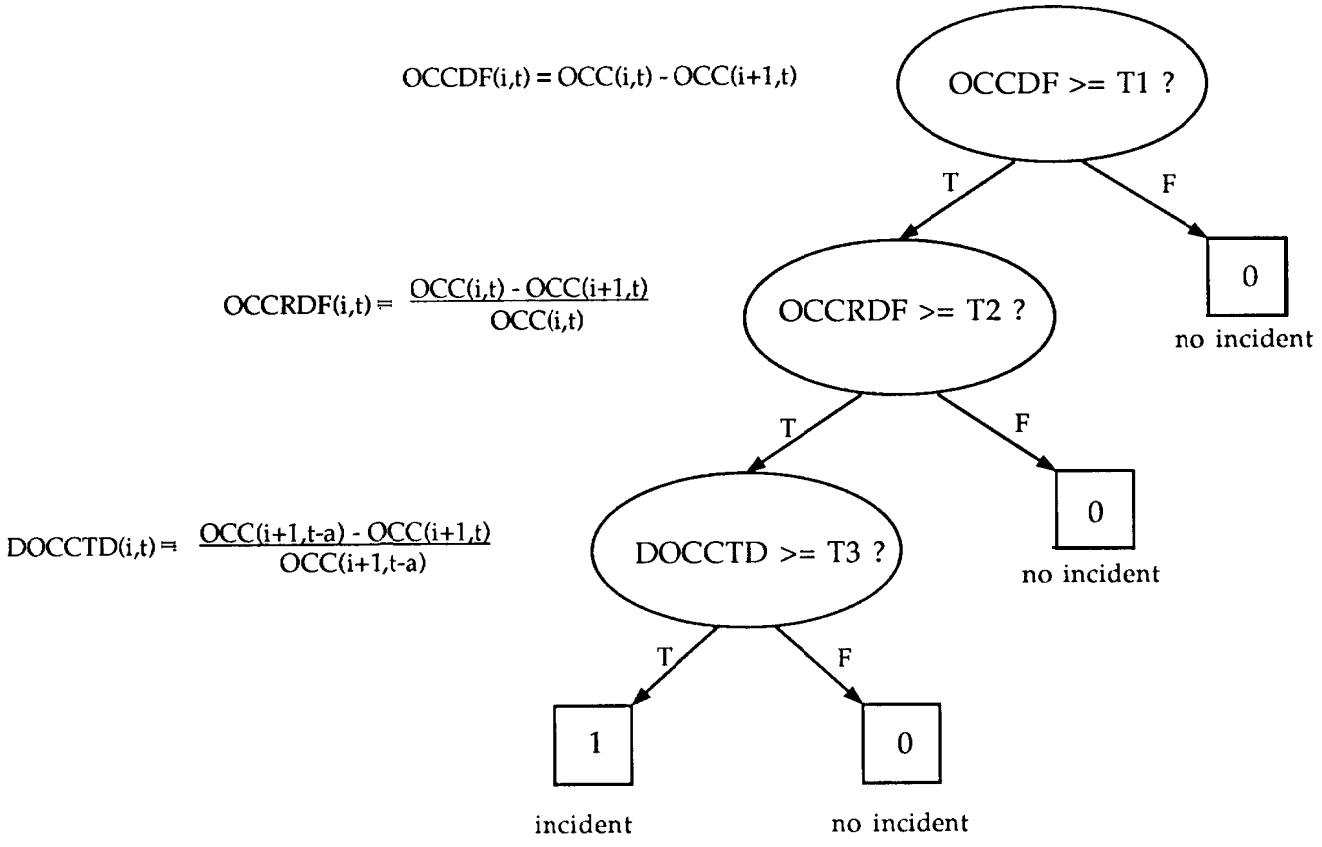

Figure 2.1 The California algorithm #2

Starting with the basic decision structure in Figure 2.1, at least eleven versions of California algorithm have been proposed and tested (Payne et al 1976). By keeping track of the conclusions deduced from the previous detection intervals, additional tests have also been added to identify the beginning and end of an incident, and to reduce false alarms caused by shock waves.

The threshold at each node of the decision tree is determined from an optimization routine which minimuzes the FAR for a given value of DR (Knobel and Helfenbein, 1976).

The current version of the California algorithm (or algorithm #8), being implemented by Caltrans on about 500 directional miles of freeways in the Los Angeles area (Arceneaux et *al* , 1989) is shown in Figure 2.2. Average occupancy across all lanes at upstream and downstream stations at 30 second intervals is used as the input. The calibration of threshold values is first performed by aggregating the data collected at freeway sections with similar geometry. The use of local data to fine tune the threshold values for each pair of adjacent stations is currently being carried out (Arcenaux *et al* , 1989).

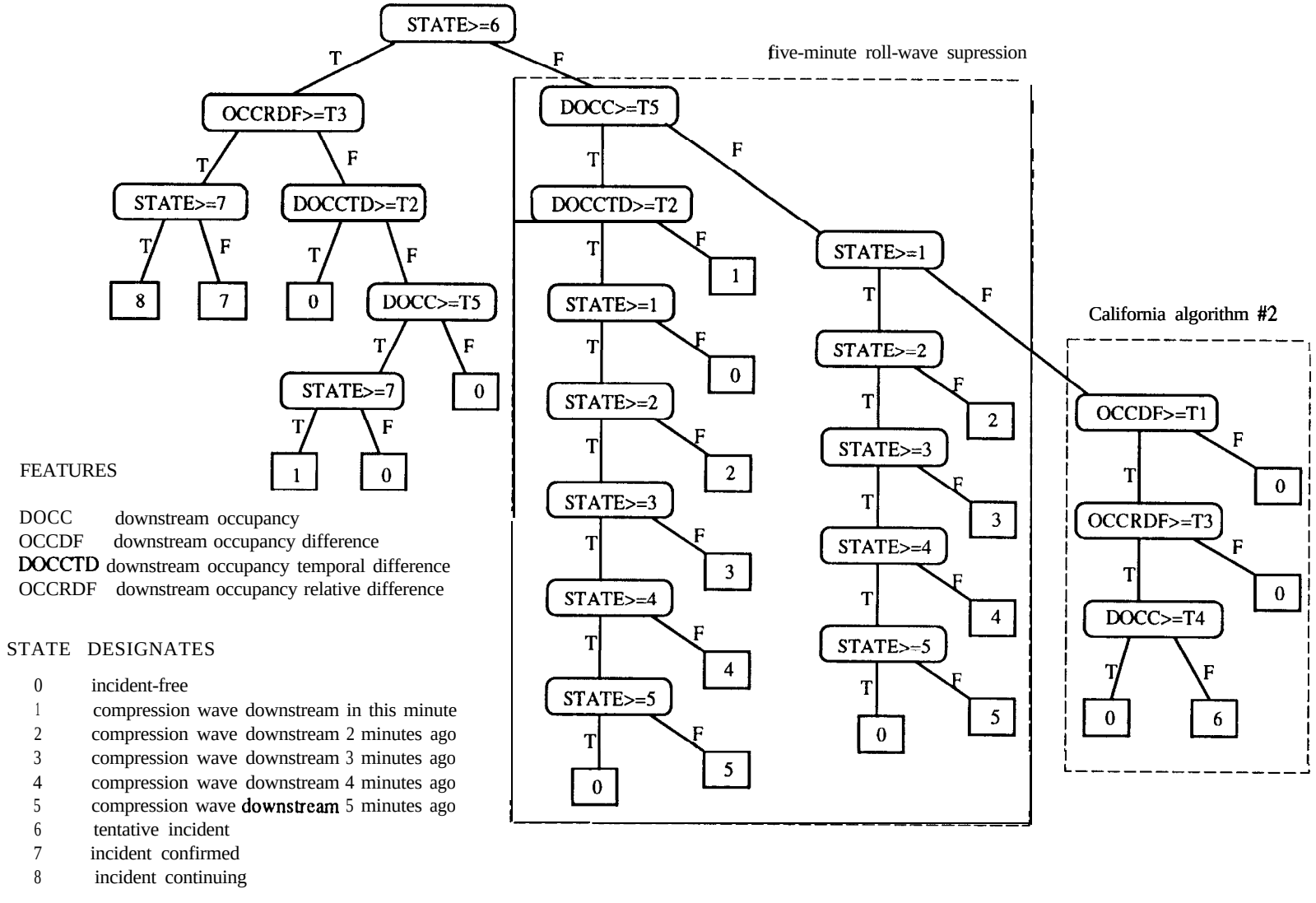

Figure 2.2 The California algorithm #8 (from Payne et al, 1976)

2.3.2 McMaster Algorithm

The McMaster algorithm has been developed relatively recently at McMaster University, Canada. This algorithm detects incident in two steps: first by detecting congestion and then by classifying the congestion as recurring or non-recurring.

The conceptual basis of this algorithm is a catastrophe theory of traffic flow describing the relationships between volume, speed and occupancy (Persaud and Hall, 1989) based on data observed at a single detector station. From this theory, the volume-occupancy plot of field detector data is divided into four regions as shown in Figure 2.3, each representing one state of traffic conditions (Gall and Hall, 1989; Aultman-Hall et al , 1991).

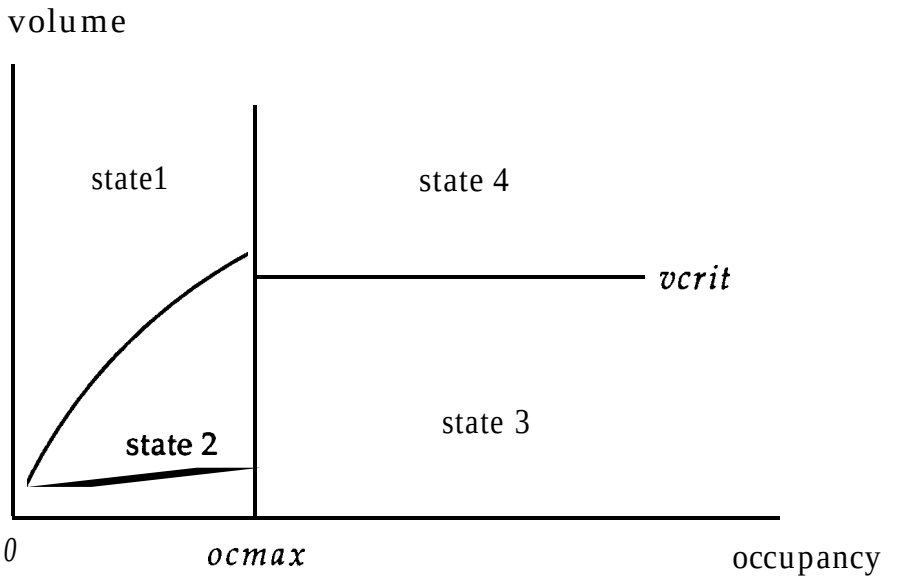

Figure 2.3 State classification of the McMaster algorithm
(from Gall and Hall, 1989)

The beginning, continuation and end of an incident downstream of a detector are identified by monitoring the movement of data points between the four regions. For example, congestion is declared if the data point moves from state 1 to state 3 and remains at state 3 for two intervals. Once an incident is suspected, the traffic states at adjacent detectors located downstream is compared to distinguish between recurring (state 4) and non-recurring (state 3) congestion. The McMaster algorithm is based on single station detection logic, using 30 second data from the median (fast) lane. A new feature in this algorithm is its capability to adjust its parameters (boundary of the four regions in the volume-occupancy plot) with weather conditions. Persaud et *al* (1990) also claim that the algorithm can detect incidents occuring immediately upstream of a station.

This algorithm has been tested with field data on the Queen Elizabeth Way and Burlington Skyway in Ontario, Canada (Persaud et *al* 1990; Aultman-Hall, 1991). Results of the tests have been encouraging. This algorithm is currently being tested on-line and is subject to further refinement.

2.3.3 Time Series and Other Algorithms

Besides the two major freeway algorithms described above, several other algorithms have been tested. A few notable ones are described briefly here.

Ahmad and Cook (1982) and Cook (1983) proposed an algorithm which uses time-series analysis of occupancy values to detect an incident. In this method, the trend of occupancy at a station over past intervals is fitted with a time-series model and a projection of the range of the value at the next time interval is made. An incident alarm is issued if the field data are two standard deviations or more away from the predicted mean.

The algorithm developed by Texas Transportation Institute (Fambro and Ritch, 1979) for detection of incidents under low volume conditions uses a pattern matching approach . The time and speed of an individual vehicle passing an upstream detector is recorded and a projection is made for a range of arrival times at the next detector station. The aggregated volume at the next station over an interval is compared with the actual volume count. An incident alarm is declared if the difference is greater than a threshold value.

The Transport and Road Research Laboratory (TRRL) in the United Kingdom was developing two algorithms in the late 1970's (Collins et al , 1979). The first algorithm, HIOCC (for HIgh OCCupancy), detects an incident when the occupancy at a detector is 100% for two consecutive seconds. The second algorithm, PATREG (for PATtern REcoGnition), uses a pattern matching technique to find the time lag in arrival patterns between two stations about 500 meters apart to estimate the space mean speed. Incidents can be detected by a significant reduction in speed.

Willsky et *al* (1980), and later Cremer and Schutt (1990) have used the Kalman filtering technique to estimate parameters describing traffic flow on a freeway segment from volume and speed measured at detector stations. The occurrence of incidents was reflected in the change in certain parameter values. Their methods, however, remain in the stage of laboratary simulation.

Recently, Blosseville et *al* (1991) reported encouraging results using image processing techniques to detect stationary vehicles from a moving traffic stream on a freeway. The algorithm detected 93% of the 133 incidents in the initial test and gave only 9 false alarms. This method is reported to work well in detecting stationary vehicles on shoulders and in the main lanes under variety of traffic conditions, and has the ability to measure queue length. The method is still under further development.

Due to the inherent positive correlation between detection rate and false alarm rate, several researchers have incorporated statistical techniques into the calibration process. Levin and Krause (1978; 1979) have tested a simple algorithm using an occupancy temporal difference from a pair of detector stations (OCCRDF) as the only input feature to classify incident or incident-free conditions. The threshold is derived from a Bayesian approach to maximize the probability of correct classification. Tsai and Case (1979) uses the maximum-likelihood principle to determine the number of intervals for persistence testing in the modified California algorithm. They have also experimented with the 'committee-machine' approach similer to the structure of neural network models.

2.4 ARTERIAL ALGORITHMS

While substential research effort has been devoted to the development of freeway incident detection algorithms, little work has been reported for arterials.

The first paper describing an attempt to develop an incident detection algorithm for arterial street systems was published by Thancanamootoo and Bell (1988). At the end of each signal cycle, thresholds for volume and occupancy at a station were computed from the means and variances of the corresponding variable collected at previous intervals by means of exponential smoothing. The actual volume and occupancy values collected at that interval were then compared with their respective thresholds. An incident alarm was declared if the upstream and downstream volumes and downstream occupancy fell below their respective thresholds and the upstream occupancy was higher than its threshold. Updating of the threshold values is suspended when an incident is confirmed until all the collected data falls to the level prior to the incident. This method has only been tested on small scale on one street block, and isolated intersections in Middlesbrough and London. Data were obtained from detectors located at the upstream end of the links. No persistence test was included.

Following the above initial attempt, Han and May (1988; 1989) also developed an algorithm and tested it briefly in a section of arterial, six blocks in length, in downtown Los Angeles. Arterial streets in downtown

Los Angeles were equiped with a network of loop detectors installed for the ATSAC (Automated Traffic Surveillance And Control) system. Detectors were located in every lane, 350 ft upstream of the stop line for every approach to an intersection. For every minute (about half the cycle time), volume and occupancy data were collected from detectors and smoothed by the following function:

$$x'(t) = (1-\beta)x'(t-1) + \beta\, x(t) \qquad \text{E-51}$$

where β=0.5, and x(t) and x'(t) represent the volume or occupancy values at time t before and after smoothing respectively. The smoothed data were passed through a module to identify detector malfunctions in which historical 1st and 99th percentile values of volume, occupancy and speed at each detector location were used to define the acceptable ranges. The traffic condition in that lane of the street section for the past minute was classified into one of several levels of warning as shown in Figure 2.4, depending on the volume and occupancy values (the data point usually fell into the area bounded by the two curves). Based on the operating conditions of the adjacent lanes at the same section and also the sections upstream of it, the type of operational problem such as lane blockage, approach blockage or arterial blockage was identified based on the logic as shown in Figure 2.5.

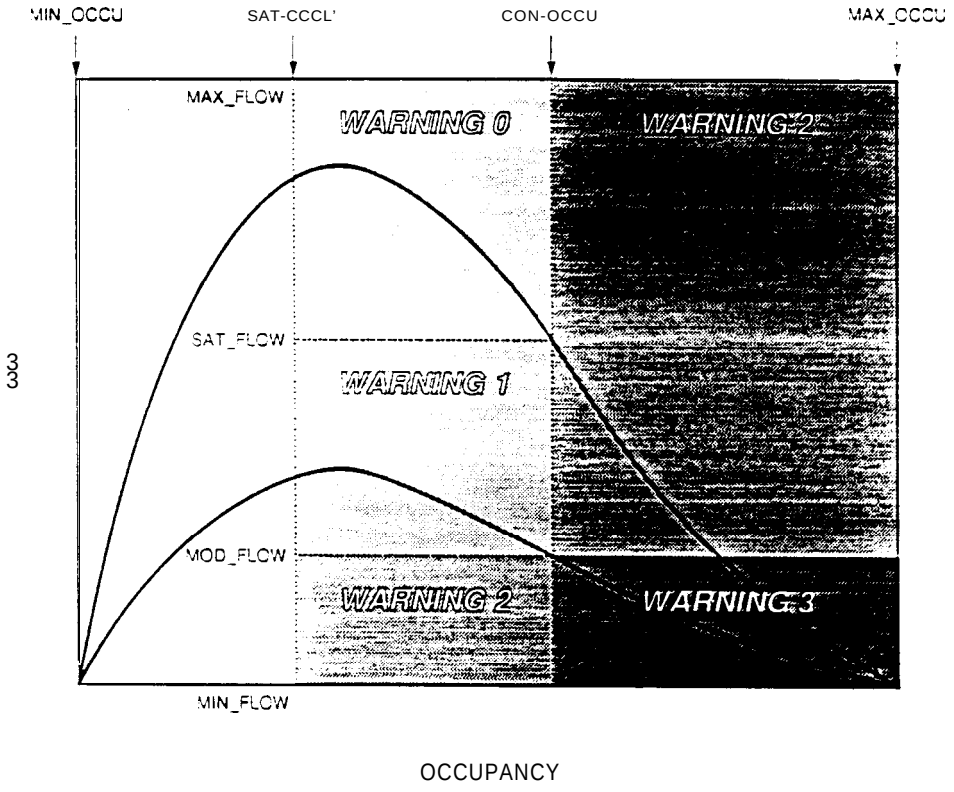

Figure 2.4 Classification of operating conditions of a single lane in an arterial street section (from Han and May, 1989)

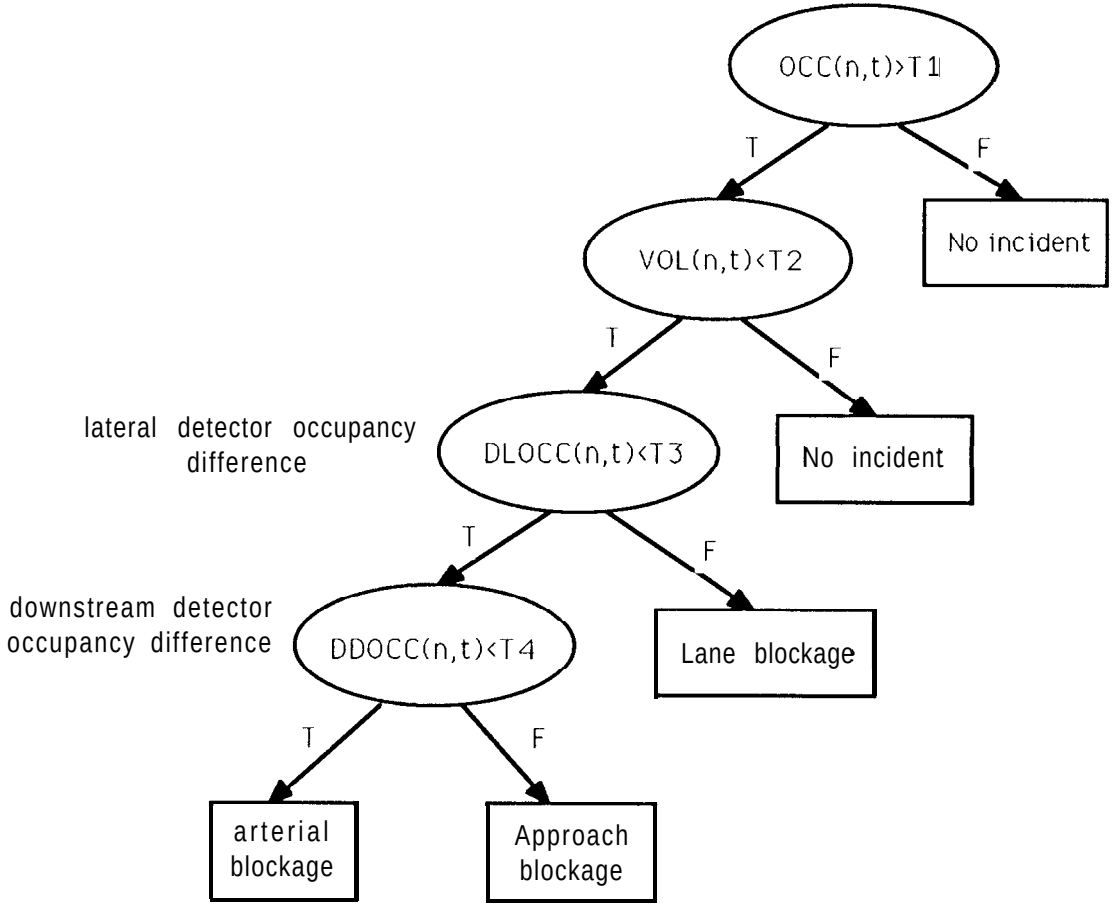

Figure 2.5 Decision tree for the identification of operational problem in an arterial street (from Han and May, 1988)

The above concept has been implemented for a section along Washington Blvd in downtown Los Angeles which contains 20 detectors. During preliminary analysis using two days of recorded data, a set of common thresholds for dividing the volume-occupancy graph was calibrated for all detectors. These thresholds were defined such that they minimised the false alarms in the test data set. They were proposed to be calibrated based on historical data and are location and time dependent (e.g. by dividing the operating hours from 6 a.m. to 8 p.m. into six periods). The threshold for classifying the nature of the operational problems in Figure 2.5 also needs to be calibrated with real data. Currently, threshold values based on the operational experience of traffic engineers in the City of Los Angeles Department of Transportation (LADOT) are being used. Preliminary research has resulted in an operational prototype program called TOPDOG written in Prolog. TOPDOG is reported to be running on-line for evaluation at the LADOT with detector and time specific thresholds.

INRETS in France is currently developing its own incident detection method for arterial intersections (Sellam et al , 1991). This project, titled INVAID, is part of the Eurorean DRIVE program, and combines real-time expert system and image processing technologies. In this method, real-time video images at four frames per second are digitized into binary form and pixels involving moving vehicles are separated from the background. Over a short period, statistics similar to occupancy are computed for each pixel. Depending on the location of the intersection each pixel represents, signal phases, and vehicle movement, rules in the expert system are applied to infer the presence of an incident.

2.5 CONCLUDING COMMENTS

Until 1988, research in incident detection has focused entirely on freeways. There have been many reports and articles on freeway incident detection algorithms since the 1970's. The majority of the studies were performed in the United States, particularly in the Los Angeles metropolitan area because of its heavily utilized freeway system. These studies resulted in the evolution of the California algorithms. However, due to the inherent positive correlation of false alarm rate with detection rate, little development of approaches exhibiting improved levels of incident detection performance has occurred. In recent years, however, researchers at McMaster University in Canada developed a new freeway algorithm and have reported substantial progress, although their algorithm has not been extensively tested with on-line data. The image processing approach used by Blosseville et *al* (1991) is also promising, but requires the installation of cameras, and other hardware at close spacing. For the freeway system in Los Angeles area which amount to about 500 directional miles, this would require major reinstrumentation efforts.

Most of the research on incident detection has concentrated on freeway incidents because traffic flow on freeways is more uniform. Freeways are designed to be limited access high (uniform) speed facilities. Extensive surveillance and control systems have been installed for freeway systems in major cities, which provide data needed for the studies. In addition, urban freeways typically have higher vehicle-miles of travel compared to arterial streets. In contrast, urban arterial streets involves a

shared right of way with several traffic streams. There is a higher degree of uncertainty in the pattern of traffic due to the close spacing of signalized intersections, turning movements, on-street parking and other curbside loading/unloading activities that may produce the same effect on traffic flow as incidents. Therefore, neural network models should first be applied to incident detection on freeways. Experience and knowledge gained in the development process will provide a better understanding for design of new incident detection model for arterial streets.

CHAPTER 3

REVIEW AND SELECTION OF NEURAL NETWORK MODELS

3.1 INTRODUCTION TO NEURAL NETWORKS

Neural networks are information processing structures based on a simplified model of the functioning of the human brain. This type of computational structure consists of many simple processing elements (PE's) or neurons interconnected with each other. The PE's correspond to the biological neurons in the human brain, while connections between the PE's symbolize the biological synapses. Within a neural network, a group of PE's receives external input signals and another group of PE's communicates the output signals to some external sources. Each PE in the network receives signals from other PE's, processes them and transmits the resulting signal to other PE's. Complex information is thus represented and processed in a parallel and distributed fashion across the network at high speed. Such networks have exhibited learning, memory, and ability to handle 'noisy' real-world data, and other significant capabilities. These networks can either be implemented in hardware as parallel computing devices, or in software simulations, as in this research.

Neural networks are suitable for problems which involve fuzzy data, when algorithmic solutions do not exist, or when the problem is complex but demands a quick solution. One of the principle applications

of neural networks is to pattern recognition problems, such as speech processing (Sejnowski and Rosenberg, 1986), character recognition (Kosko, 1987), and image classification (Lo and Bavarian, 1991). Such techniques have only recently been applied to civil engineering problems. Examples of such applications are project cost estimation (Moselhi et *al*, 1991), structural design (Gan and Liu, 1991; Hajela et *al*, 1991) and crack depth evaluation (Ramirez and Arghya, 1991). In transportation engineering, neural networks have also begun to be used, although still in limited domains, in processing of pavement images for distress classification (Kaseko and Ritchie , 1991), traffic assignment (Akamatsu et *al*, 1990) and traffic signal setting (Nakatsuji and Terutoshi, 1990).

3.2 CHARACTERISTICS OF NEURAL NETWORKS

The basic components of neural networks are PE's and their connections. As opposed to the massive structures in the human brain, PE's in a typical artificial network are organized into layers as shown in Figure 3.1. Based on the direction of the flow of information, the layer of PE's that receives input signals from external sources is called the input layer, while the layer of PE's that transmits signals to external sources is called the output layer. Layers of PE's arranged in between the input and output layers are called hidden layers. Depending on the topology, some networks have no hidden PE's while others may have one or more hidden layers. A set of signals which is simultaneously fed to the PE's in the input layer, such as $(x_1, x_2, \ldots, x_m)$ in Figure 3.1, is called the input

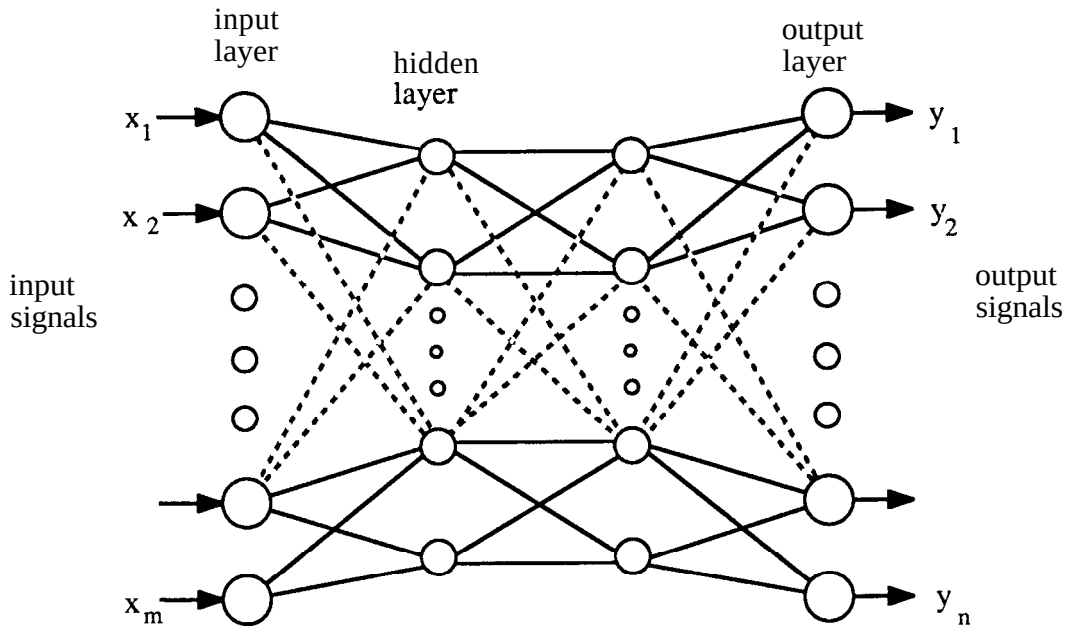

Figure 3.1 Topology of an artificial neural network

vector. The corresponding output, $(y_1, y_2, \ldots, y_n)$, is referred to as the output vector.

Associated with each PE is an activation function which transforms the weighted sum of its input signals into the PE's output. The input-output signal of each PE is sometimes referred to as short term memory. Common activation functions are in the form of a step function, linear function and sigmoid (or logistic) function.

Connections generally exist between the PE's in two adjacent layers, although some networks have connections between the PE's in the same layer. Each connection carries a weight which steps up/down the signal transmitted between the two PE's. Weights that stored in the connections are permanent and are therefore also known as long term memory.

A neural network needs to be trained so that it can produce the desired outputs when presented with relevant inputs. Given a neural network with specified layers of PE's, connections and activation functions, the objective of training is to let the network adapt the correct connection weights and parameters in the activation functions so as to perform the desired mapping between the input and output patterns. Typically, a large number of input-output vector pairs, which represent the population of the pattern space, is used for training. Training is accomplished by sequentially applying input vectors and adjusting connection weights according to a certain pre-determined procedure. Upon the repetitive application of the training vectors, weights will

normally converge to values such that the network learns and memorizes the association between each pair of input-output patterns.

Training algorithms may be categorized as supervised or unsupervised. In supervised training, each pair of input and output vectors is presented to the network. From the existing connection weights and other parameters, the output produced by the net is compared with the desired output. Corrections to the weights are made to reduce the output error. One example of supervised training is the popular backpropagation algorithm used for multi-layer feed-forward neural networks. In unsupervised learning, no comparison is made between the network's output and the desired output. The network adjusts the weights by itself according to some specified procedure such that similar input vectors will produce the same or consistent output patterns. Essentially, this training process extracts properties of the training vectors and groups similar vectors into classes. The self-organizing feature map developed by Kohonen (1982) is a typical neural net that uses unsupervised training.

3.3 CLASSIFICATIONS OF NEURAL NETWORK MODELS

According to Rumelhart and Zipser (1986), neural networks can be categorized into four main types based on their functions. They are (1) auto associator, (2) pattern associator, (3) classification paradigm, and (4) regularity detector. The auto associator 'stores' a set of patterns in the net.

When an input pattern similar to the format of a stored pattern is presented to the network, the net 'retrives' the stored pattern which is closest to the input. An example of an autoassociator is the Hopfield net (Wasserman, 1989). The pattern associator performs the mapping of the input patterns to the output patterns. Given a marginally distorted input vector, the network is capable of recognizing the closest desired input vector and produces the corresponding output. The bidirectional associative memory developed by Kosko (1987) falls into this category. Classification paradigms, such as the multi-layer feed-forward net (Rumelhart et *al* , 1986) and learning vector quantization (Kohonen, 1990), are modified versions of the pattern associator which perform many-to-few mappings. This type of network classifies many different input patterns into relatively few output patterns. The regularity detector is designed to automatically capture the most salient features of the input patterns. It serves to identify important features that separate the input patterns into classes. Kohonen's self-organizing feature map (Kohonen, 1982) is the representative paradigm in this category.

The classification paradigm and regularity detector have the highest potential among the four types of neural networks to be used for incident detection. In this problem, numerous patterns in traffic flow data must be classified by the classification paradigm to identify incidents, shock waves or other operational problems. The regularity detector might help to identify the characteristics of the detector data during incident conditions.

3.4 MODEL SELECTION

More than 26 types of neural network are listed in Table 3.1 (Simpson, 1990). Since different networks have been designed to perform tasks suitable for their problem domains, they usually receive certain forms of input and produce designated forms of output. Characterized with each of these networks is their training algorithm for its parameters. In the table, some important properties of the neural networks are listed. These properties are, in order from left to right,

no. of PE layers
form of input and output signals
mode of operation
off-line or on-line learning capability
supervised or unsupervised training

The signals presented / produced by the neural networks come in two forms: analog and digital. Analog signals are continuous, and digital signals are discrete with value of either 0 or 1. The operation of the neural net can be discrete or continuous. A discrete network will produce an output vector once every input vector is presented. In a continuous mode of operation, the net can receive continuous signals that vary with time, and simultaneously give output signals. Few networks are designed to operate in continuous mode. The capability of the neural net to learn with on-line data is an important aspect of its properties. Some applications require the network to self-adapt to the operating conditions

Table 3.1 Characteristics of different neural network models

network name	layer of PE's	type of signals* input	type of signals* output	mode of operation**	method of training on/offline	method of training un/supervised
Additive Grossberg (AG)	1	A	B	C	online	unsupervised
Shunting Grossberg (SG)	1	A	B	C	online	unsupervised
Adaptive Resonance Theory 1 (ART1)	2	B	B	D & C	online	unsupervised
Adaptive Resonance Theory 2 (ART2)	2	A	B	D & C	online	unsupervised
Discrete Autocorrelator (DA)	1	B	B	D	offline	unsupervised
Continuous Hopfield (CH)	1	A	A	D	offline	unsupervised
Bidirectional Associative Memory (BAM)	2	B	B	D	online	unsupervised
Adaptive Bidirectional Associative Memory (ABAM)	2	A	A	C	offline	unsupervised
Temporal Associate Memory (TAM)	2	B	B	D	offline	unsupervised
Leraning Matrix (LM)	2	A	A	C	offline	unsupervised
Drive Reinforcement	2	A	A	D	offline	unsupervised
Sparse Distributed Memory (SDM)	3	B	A	D	offline	unsupervised
Linear Associate Memory (LAM)	2	A	A	D	offline	unsupervised
Optimal Linear Associative Memory (OLAM)	2	A	A	D	offline	unsupervised
Fuzzy Associative Memory (FAM)	2	A	A	D	offline	unsupervised
Learning Vector Quantization (LVQ)	2	A	A	D	offline	unsupervised
Counterpropagation (CPN)	3	A	A	D	offline	unsupervised
Self-Organizing Feature Map (SOFM)	2	A	B	D	offline	unsupervised
Brain-State-in-a-Box (BSB)	1	A	A	D	offline	supervised
Fuzzy Cognitive Map (FCM)	1	A	A	D	offline	supervised
Multi-Layer Feed-Forward Net (MLF)	23	A	A	D	offline	supervised
Adaline/Madaline	>1	A	B	D	offline	supervised
Boltzman Machine (BM)	3	B	B	D	offline	supervised
Cauchy Machine (CM)	3	B	B	D	offline	supervised
Adaptive Heuristic Critic (AHC)	2 or 3	A	B	D	offline	supervised
Associate Reward-Penalty (ARP))	2	A	B	D	offline	supervised
Avalanche Matched Filter (AMF)	2	A	A	D	offline	supervised

* type of signals: A=analog, B=binary
** mode of operation: C=continuous, D=discrete

and others may want the net to learn with prior data. For application in freeway incident detection, the following factors were considered in selecting the type of network:

1. Mapping of Patterns. The nature of incident detection requires mapping of large amount of information, such as the volume, occupancy and speed from several loop detectors which form numerous combination of values, into a few outcomes. Therefore, (a) pattern associators, such as bidirectional associative memory, adaptive bidirectional associative memory, temporal associative memory and fuzzy associative memory are not suitable; and (b) networks which perform auto association, such as additive Grossberg, shunting Grossberg, discrete autocorrelator, continuous Hopfield, brain-state-in-a-box and fuzzy cognitive map, are not useful for this purpose. The network needs to store hetero-associative patterns of input and output vectors. There needs to be at least two layers of PE's. In addition, the network should have the ability to perform non-linear mapping. This constraint excludes linear associated memory, optimal linear associative memory and adaline/madaline.

2. Input data. Detector data, such as volume and occupancy, are continuous variables. Hence the input PE's in a network would preferably be able to receive analog (continuous) values. Networks which use binary input such as adaptive resonance theory 1, discrete autocorrelator, bidirectional associative memory, temporal associative memory, sparse distributed memory, Boltzman machine and Cauchy

machine, are not suitable. Networks designed for analog input may also accept 0 or 1 values.

3. Off-line Training. Training patterns should be selected with care so that network parameters trained for typical patterns will be stored. In this application, on-line learning is typically not possible because any occurrence of incident needs to be verified by the traffic operational personnel, and this process involves some delay. Training therefore has to be conducted off-line.

4. Discrete Operation. Since the data collection interval at Caltrans detector stations is 30 seconds, the neural network must initially follow the current practice in accepting measured traffic parameters at multiples of 30 seconds. The input pattern will thus be presented to the network at discrete time intervals.

With the above constraints imposed by the problem domain, the remaining neural networks which satisfy the conditions are the multi-layer feed-forward neural network, learning vector quantization, self-organizing feature map, counterpropagation, adaptive heuristic critic, avalanche matched filter and drive reinforcement. The adaptive heuristic critic and drive reinforcement nets have not been implemented in many studies and the limited literature on these two networks was not easily accessible. Among the remaining nets, the multi-layer feed-forward neural net is the most frequently investigated model. The multi-layer feed-forward neural net was therefore selected for this study.

CHAPTER 4

THE MULTI-LAYER FEED-FORWARD NEURAL NETWORK

4.1 INTRODUCTION

The neural network model adopted for this study is the multi-layer feed-forward neural net (MLF). It is also known as the backpropagation network or multi-layer perceptron. The concept of the perceptron was first introduced by Rosenblatt (1962), and extensive demonstrations of the training and application of this type of neural network model have been reported by Rumelhart et al (1986). The operation of the MLF and its training algorithm are described in the following subsections.

4.2 TOPOLOGY AND OPERATION

The MLF used in our investigation consists of PE's arranged in three layers: the input layer, a hidden layer and the output layer. Interconnected PE's in adjacent layers have connection weights w_{ij} and v_{jk}, as shown in Figure 4.1.

When a set of input values (a,, a_2, . . . , a,} is presented to the network, each of the PE's in the hidden layer receives the weighted sum of all

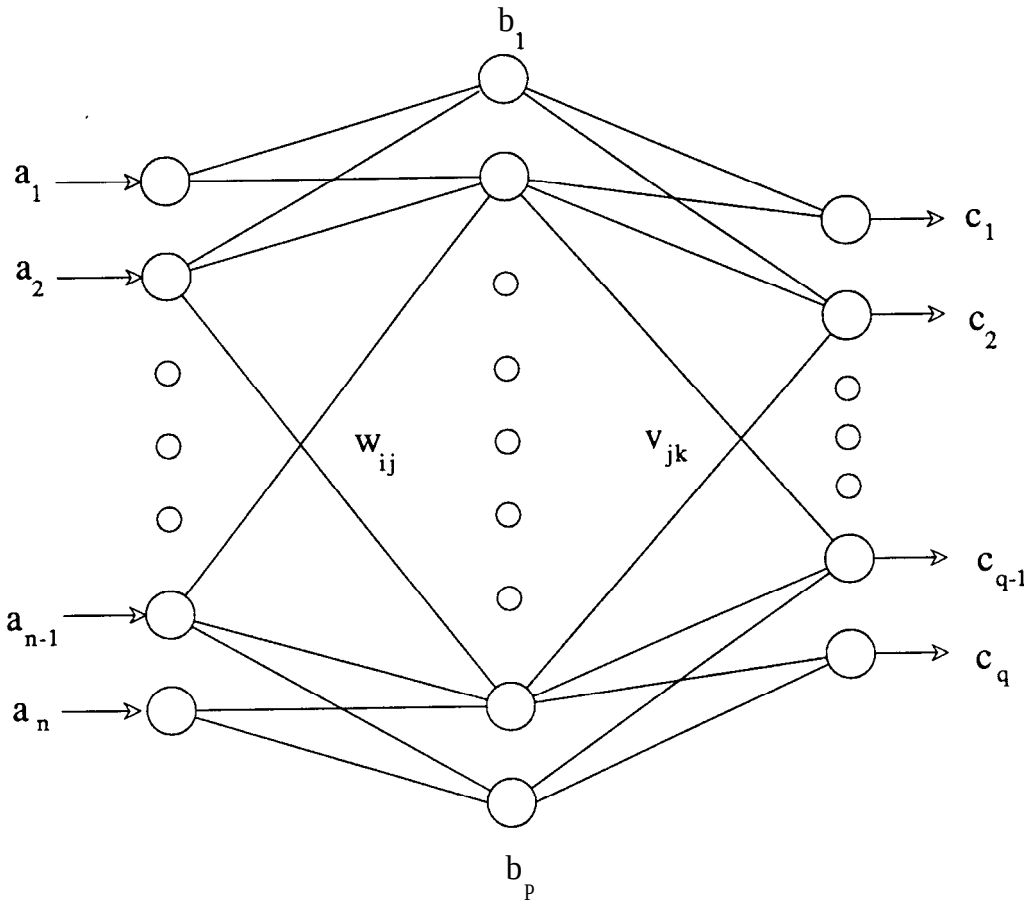

Figure 4.1 Topology of the multi-layer feed-forward neural network

the input values. This is passed through a transfer function $f(\ )$ and produces an output value b_j . Mathematically:

$$b_j = f\left(\sum_{i=1}^{n} w_{ij} a_i + \theta_j\right) \qquad \forall j=1,\ldots,p \qquad [4.1]$$

where the threshold value of the jth PE in the hidden layer, θ_j , adjusts the magnitude of the output, and $f(\ )$ is a sigmoid (or logistic) function which takes the following form:

$$f(x) = \frac{1}{1 + e^{-x}} \qquad [4.2]$$

The same process is repeated for PE's in the output layer, where:

$$c_k = f\left(\sum_{j=1}^{p} v_{jk} b_j + \phi_k\right) \qquad \forall k=1,\ldots,q \qquad [4.3]$$

Again, ϕ_k is the threshold value of the kth PE in the output layer. The vector $\{c_1, c_2, \ldots, c_q\}$ is commonly referred to as the output vector.

4.3 BACKPROPAGATION TRAINING

Proper values of connection weights, w_{ij} and v_{jk}, and thresholds, θ_j and ϕ_k , have to be assigned to the MLF for it to perform its intended function. That is, given an input vector which consists of the upstream and downstream detector data, the MLF must be able to produce the correct output vector which indicates incident or incident-free conditions. The

process of assigning these network parameters is called training. During the training process, input and desired output vectors representing the population of the data to be classified by the network are used as training examples. Each corresponding input and desired output vector in the training data set is applied to the network. For each input vector, the output values produced by the network are compared with the desired results, and adjustments are made to the network parameters until the magnitude of the output error becomes acceptable. For the MLF, the backpropagation procedure described by Rumelhart et al (1986) is used in the training. This approach seeks to implement a gradient descent on an error function of the network's output. The detailed training procedure is as follows:

(1) Initially assign small random numbers to all the weights, w_{ij} and v_{jk}, and threshold values, θ_j and ϕ_k.

(2) Present an input vector to the network. The output values of the network are computed from the steps discussed in Section 4.2. The errors in the output signals are then computed and propagated backwards to the v_{jk} and then w_{ij} layers for the updating of these weights.

(3) For the kth PE in the output layer, the error δ_k is first calculated as:

$$\delta_k = c_k (1 - c_k)(c_k' - c_k) \qquad \forall k=1,...,q \qquad [4.4]$$

where c_k' is the desired output value. The magnitude of the change in weight for v_{jk} after the presentation of the input vector m is then:

$$\Delta v_{jk}(m) = \eta\, b_j\, \delta_k + a\, \Delta v_{jk}(m-1) \qquad \forall j=1,\ldots,p;\ \forall k=1,\ldots,q \qquad 14.51$$

The first expression on the right hand side is a correction based on the current error where η is the learning rate. The magnitude of this adjustment is directly proportional to the value of the output signals in the hidden layer (b_j), the error of the network's output ($c_k'-c_k$) and the gradient of the sigmoid function ($c_k(1-c_k)$). The second expression on the right hand side is the momentum term from the previous adjustment which is added to increase the rate of convergence with small learning rates, and to dampen possible oscillations in the weight changes. The *a* value is referred to as the momentum rate. The learning and momentum rates are usually between 0 and 1.

(4) To adjust the w_{ij}, the error ζ_j of the jth PE in the hidden layer is computed by using the sum of the δ_k's in the output layer weighted by the v_{jk}'s, using their values before the correction in step (3):

$$\zeta_j = b_j\,(1-b_j)\sum_{k=1}^{q}(v_{jk}\delta_k) \qquad \forall j=1,\ldots,p \qquad [4.6]$$

$$\Delta w_{ij}(m) = \eta\, a_i\, z_j + a\, \Delta w_{ij}(m-1) \qquad \forall i=1,\ldots,n;\ \forall j=1,\ldots,p \qquad 14.71$$

(5) Thresholds for each PE in the hidden and the output layers are each updated by:

$$\Delta\phi_k(m) = \eta\, \delta_k + a\, \Delta\phi_k(m-1) \qquad \forall k=1,\ldots,q \qquad [4.8]$$

$$\Delta\theta_j(m) = \eta\, \zeta_j + a\, \Delta\theta_j(m-1) \qquad \forall j=1,\ldots,p \qquad [4.9]$$

(6) The sum of squared errors (SSE) of the output layer is also computed:

$$SSE(m) = \sum_{k=1}^{q}(c_k'-c_k)^2 \qquad [4.10]$$

(7) Repeat steps (2) to (6) for all the training vectors. After the presentation of all the training vectors, the sum of all SSE(m)'s is used to check the convergence of the network parameters. The weights are said to have converged if the total SSE for the entire training set (total SSE in one training cycle) is less than a specified value, in which case the training is stopped. Otherwise, steps (2) to (6) are repeated for the next cycle of training with the same set of training vectors.

When the total sum of squared errors is within some tolerance limit, the network is said to have learned and memorized the mapping between the input and output vectors.

CHAPTER 5

PRELIMINARY CASE STUDY

5.1 INTRODUCTION

This chapter presents a preliminary investigation of a neural network model for incident detection on freeways. In this investigation, synthetic detector data were generated by Monte-Carlo simulation and used to train a neural network model. Microscopic simulation was performed to model traffic operations, and the occurrence of an incident in an actual freeway testbed. The trained neural net model was then applied to the simulation data to illustrate the process of detecting an incident.

5.2 STUDY FRAMEWORK

In this study, continuous sections of freeway were divided into shorter segments, based on the availability of the data and the level of detail of the investigation, so that traffic conditions in each segment could be analyzed. To determine the presence or absence of an incident at a particular location based on surveillance data collected from existing loop detectors, the freeway was divided into sections separated by adjacent

detector stations. In the metropolitan Los Angeles area, the spacing between adjacent stations varies from about 0.25 to over 1.0 mile, and is typically 0.50 miles. Most stations have at least 3 loop detectors installed across the through travel lanes. Caltrans currently collects data from these stations at 30 second intervals. Detector data are in the form of lane-specific volume and occupancy values (occupancy is the proportion of time a detector is 'activated' by vehicles), although some stations are also able to provide average speed.

This effort focused on developing a neural network model that could accept detector data from the upstream and downstream end of a freeway section at 30 second intervals and classify the traffic condition in the section, at the past 30 second interval, into one of four states:

State 1: Incident-free
State 2: Beginning of queue
State 3: Queued section
State 4: End of queue

State 2 corresponds to a suspected incident location (or, after the occurrence of an incident, the front of a dissipating queue). State 3 corresponds to a queue extending over the entire section, and State 4 the location of the tail of the queue. For any freeway section, the onset of an incident may be detected by a change from State 1 to State 2 and for continuous sections, the existence and extent of an incident queue are characterized by the presence of the last three states.

5.3 GENERATION OF TRAINING DATA

For demonstration purposes in this initial investigation, a somewhat simplified approach to neural network training was implemented using synthetic data. The network parameters were trained with simulated detector data derived from a plot of 30 second average volume and occupancy such as that presented in Figure 5.1, which shows 400 data points generated for one detector station. This volume-occupancy plot is based on the envelop of data points reported by Persaud et al (1990) for the Burlington Skyway in Ontario, Canada. In Figure 5.1, the region bounded by occupancies greater than 0.26 and volumes less than 1000 vehicles per hour per lane (vphpl) is designated in this study as the congested region, while the remaining region is designated as the uncongested region. The critical occupancy of 0.26 is based on the value used by Persaud et *al* (1990) while the critical volume of 1000 vphpl is arbitrarily selected to demonstrate the capability of the MLF in performing piecewise linear classification of the data. For each freeway section, 30-second average volume and occupancy across all lanes were generated by Monte Carlo methods for the upstream and downstream stations, respectively. The desired states of traffic conditions at the freeway section were assigned based on the corresponding regions for the data points at the upstream and downstream stations according to Table 5.1.

Based on the above approach, 1000 training vectors (each comprising the volume and occupancy at the upstream and downstream stations and the desired states) were generated and used in the training.

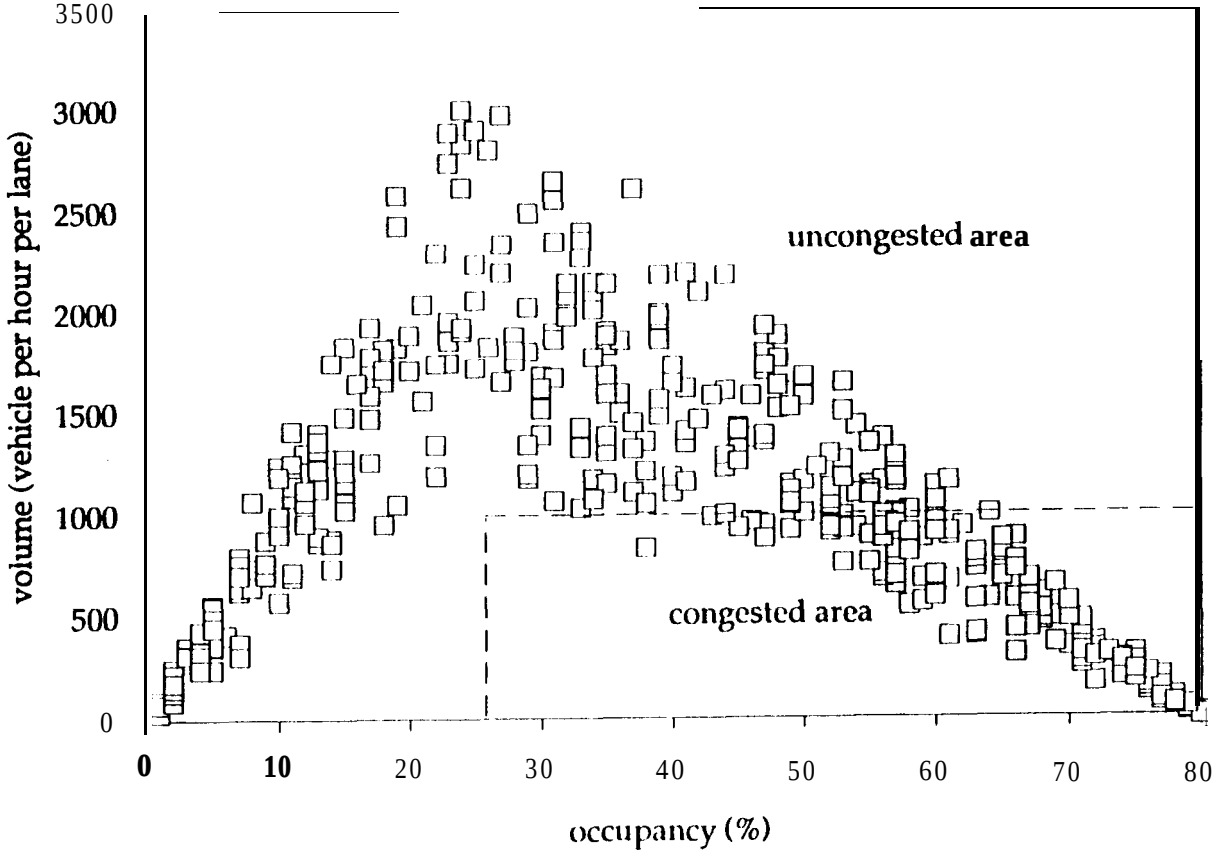

Figure 5.1 Simulated 30-second detector data for case study

Table 5.1 Assignment of desired states for training of MLF in preliminary case study

Upstream Station	Downstream Station	Desired State
unconges ted	unconges ted	1
congested	unconges ted	2
congested	congested	3
unconges ted	congested	4

5.4 RESULTS OF MULTI-LAYER FEED-FORWARD NEURAL NETWORK TRAINING

The volume and occupancy values at the upstream and downstream stations formed the basic input variables to the MLF. All input values were scaled to between 0 and 1 to avoid scale bias of any input variable in dominating the input vector. Therefore, the volume/maximum flow ratio was used instead of volume. The maximum flow of the freeway was taken as 3500 vphpl which is the maximum value in Figure 5.1. Four output PE's were used in the network, with each PE corresponding to one of four states. The desired output vectors for the four states were coded as:

State 1: { 1, 0, 0, 0 }

State 2: { 0, 1, 0, 0 }

State 3: { 0, 0, 1,0 }

State 4: { 0, 0, 0, 1 }

Other input features such as the spatial difference in occupancy, spatial difference in volume/maximum flow ratio and volume/maximum flow/occupancy (which is an indicator of the time-mean speed), were also considered. In addition, the number of PE's in the hidden layer was varied between 6 and 12.

It was found that the four input features based on the occupancy and the volume/maximum flow ratio at the upstream and downstream

stations consistently gave better results compared with other combinations of input variables. The network consisting of 10 hidden PE's gave the minimum overall error. The rate of convergence of the network parameters during training can be seen from the plot of total sum of squared errors (between the output PE's and the desired output) in Figure 5.2. After 5000 iterations of training, the total sum of squared errors dropped to 0.085, which corresponds to an average error of $2x10^{-5}$ between the network output and the desired output of a PE for each training vector. With these network parameters, 100% recall of all the training vectors was achieved. In other words, the MLF was able to correctly memorize and classify the state of each MLF required approximately 3.5 hours on a Sun SPARCstation 1.

In a subsequent test, application of this trained MLF to an independently simulated set of 1000 patterns yielded an overall recall of 99.1% for all four States, and 99.5% for State 2, the incident State of primary interest in detecting. In this test, no other patterns in States 1, 3 or 4 were misclassified as State 2, so a false alarm rate of 0% can be reported for this demonstration. If future aaaplication of neural networks replicate such performance in practice, a breakthrough in freeway traffic management may be achieved.

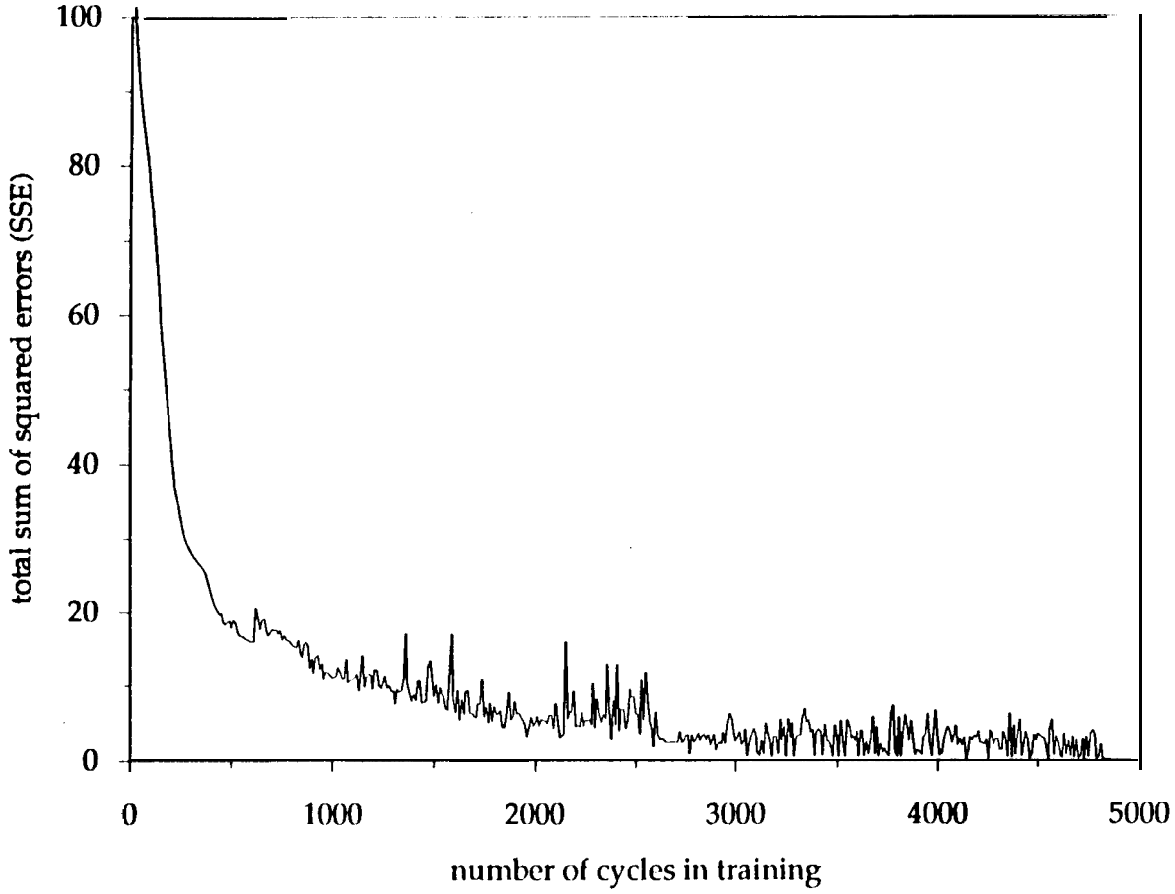

Figure 5.2 Error reduction during neural net training for case study

5.5 OFF-LINE CASE STUDY

The applicability of the MLF to detect an incident is best demonstrated by testing it on an actual freeway testbed, using on-line detector data. A 6 mile section of the eastbound SR-91 Freeway in Orange County, California was selected as the testbed for this study. However, the present study uses data generated from a microscopic simulation of traffic operations at the testbed to illustrate the process of detecting an incident.

The most recent version of the well-known INTRAS (Integrated Traffic Simulation) model (Wicks and Andrews, 1980) was obtained from the U.S. Federal Highway Administration and made operational for the testbed segment, replicating the actual geometry and detector station locations and configurations of the freeway, as shown in Figure 5.3. INTRAS permits a microscopic simulation of freeway operation under incident conditions, and outputs detector occupancies and volumes as needed for incident detection purposes. Under conditions of heavy traffic (2000 vphpl), a 5 minute incident from 300 to 600 seconds which blocked two lanes within section 9 (as shown in Figure 5.3) was simulated. Individual loop detector output was extracted at every 30 second interval and converted into each station's average occupancy and volume to be used by the MLF.

The traffic condition states that were classified by the MLF are presented in Table 5.2. The neural network model correctly detected the incident at 360 seconds, and without any false alarms. The 60-second time

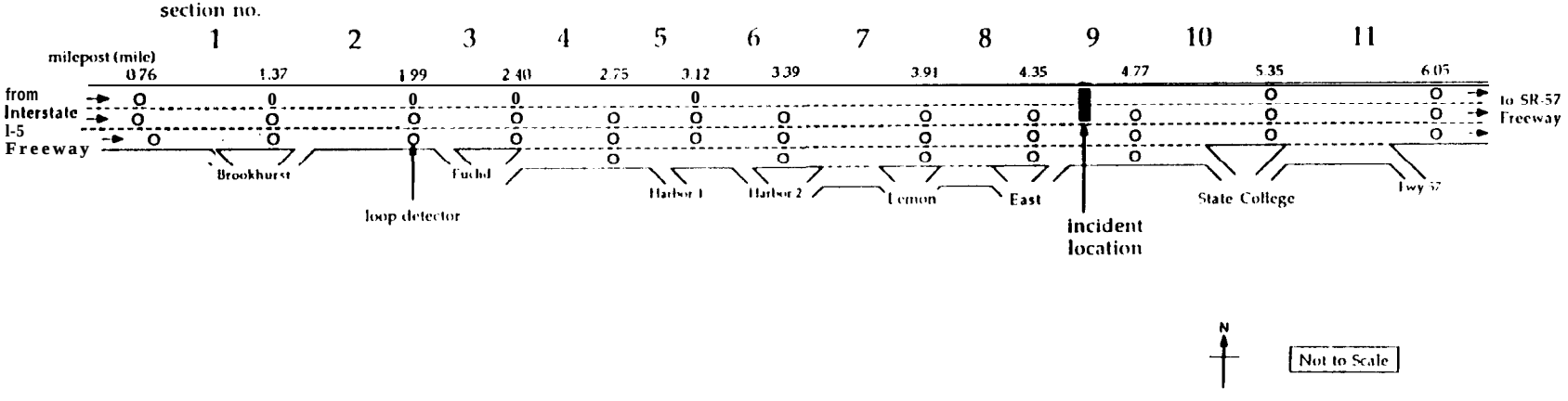

Figure 5.3 Schematic diagram of SR-91 Freeway testbed (eastbound)

Table 5.2 Classified traffic states of simulation data for preliminary case study

time (sec)	freeway section 1	2	3	4	5	6	7	8	9	10	11
30	1	1	1	1	1	1	1	1	1	1	1
60	1	1	1	1	1	1	1	1	1	1	1
90	1	1	1	1	1	1	1	1	1	1	1
120	1	1	1	1	1	1	1	1	1	1	1
150	1	1	1	1	1	1	1	1	1	1	1
180	1	1	1	1	1	1	1	1	1	1	1
210	1	1	1	1	1	1	1	1	1	1	1
240	1	1	1	1	1	1	1	1	1	1	1
270	1	1	1	1	1	1	1	1	1	1	1
300	1	1	1	1	1	1	1	1	1	1	1
330	1	1	1	1	1	1	1	1	1	1	1
360	1	1	1	1	1	1	14	2	1	1	
390	1	1	1	1	1	1	14	2	1	1	
420	1	1	1	1	1	1	1	4	2	1	1
450	1	1	1	1	1	1	14	2	1	1	
480	1	1	1	1	1	1	14	2	1	1	
510	1	1	1	1	1	1	14	2	1	1	
540	1	1	1	1	1	1	1	4	2	1	1
570	1	1	1	1	1	1	14	2	1	1	
600	1	1	1	1	1	1	1	4	2	1	1
630	1	1	1	1	1	1	1	4	3	2	1
660	1	1	1	1	1	1	1	4	3	2	1
690	1	1	1	1	1	1	4	3	2	1	1
720	1	1	1	1	1	14	3	2	1	1	
750	1	1	1	1	1	14	3	2	1	1	
780	1	1	1	1	1	14	3	2	1	1	
810	1	1	1	1	1	14	3	2	1	1	
840	1	1	1	1	1	1	4	3	2	1	1
870	1	1	1	1	1	14	2	1	1	1	
900	1	1	1	1	1	14	2	1	1	1	
930	1	1	1	1	1	14	2	1	1	1	
960	1	1	1	1	14	4	2	1	1	1	
990	1	1	1	1	14	2	1	1	1	1	
1020	1	1	1	1	14	2	1	1	1	1	
1050	1	1	1	1	14	2	1	1	1	1	
1080	1	1	1	14	4	2	1	1	1	1	
1110	1	1	1	14	2	1	1	1	1	1	
1140	1	1	1	14	2	1	1	1	1	1	
1170	1	1	1	14	2	1	1	1	1	1	
1200	1	1	1	1	1	1	1	1	1	1	1
1230	1	1	1	1	1	1	1	1	1	1	1
1260	1	11	11	11	1	11	1				
1290	1	1	1	1	1	1	1	1	1	1	1
1320	1	1	1	1	1	1	1	1	1	1	1
1350	1	1	1	1	1	1	1	1	1	1	1

lag is due to the time for the congestion queue to propagate to the upstream station after the onset of the incident. The classified states also showed the extension and dissipation of the vehicle queue during, and after the removal of, the incident.

For comparison, the California Algorithm #8, which is currently being used in the Los Angeles freeway system, was applied to the same incident data set using the threshold set #1 calibrated by Payne et *al* (1976). This threshold set previously resulted in the highest detection rate and shortest time to detection for this algorithm. The California Algorithm #8 declared the incident at 480 seconds. Besides having an additional 2 minutes of delay in detecting this incident compared with the MLF, the California algorithm was not capable of showing the queue condition at the testbed during the period of interest.

Despite limitations associated with features of the training process and the questionable transferability of the training data, we believed that the performance of the MLF in this test was encouraging.

5.6 CONCLUDING COMMENTS

This initial investigation has shown the MLF to be of potential use for incident detection. To further develop a software prototype neural network model for detecting incidents at the testbed, simulation runs were performed with INTRAS to generate detector data under different incident

conditions. The advantages of using simulation data for training and this early stage of testing are two-fold: (1) it allows the generation of many types of incidents under different flow conditions for which actual data would be extremely difficult to obtain; (2) the duration, location and nature of an incident is deterministic, enabling assignment of the desired states of the output vectors for training and evaluation of the model.

CHAPTER 6

DESIGN OF NEURAL NET PARADIGMS

6.1 INTRODUCTION

The California algorithm is based on changes in occupancy values at upstream and downstream stations. In this algorithm, upstream and downstream occupancy are combined into several features. Two primary features used to trigger incident alarms are the occupancy difference between the upstream and downstream stations (OCCDF), and downstream occupancy temporal difference (OCCRDF). To trigger an incident alarm in the California algorithm, the incident queue has to be propagated to the upstream detector, thereby increasing the upstream occupancy. The McMaster algorithm also uses an increase in upstream occupancy to detect an incident.

The neural net in the preliminary case study discussed in the previous chapter made use of both upstream and downstream data to classify traffic states in a freeway section. From experience gained in the case study, it is necessary to use additional inputs based on the detector data to reduce the time-to-detection. The detector output during an incident was therefore studied, and several possible input features to the neural network model were proposed.

In this chapter, we first discuss the pattern of detector data during the beginning of incidents. From this, input features to the MLF's were selected. The output features of the nets were next decided based on the number of patterns to be classsified.

6.2 PATTERNS OF DETECTOR DATA DURING INCIDENTS

When a typical incident occurs, the incident queue propagates upstream at a speed depending on the in-coming traffic volume and the reduced capacity. Downstream of the incident, the time lag for a reduction of volume and occupancy to appear in the detector data can be calculated based on the average stream speed before the incident. For a lane-blocking incident located at the mid-point between two detector stations, the effect of the incident will appear first in the downstream station's data. Making use of primarily downstream station data for inputs to the MLF's could therefore reduce the time to detect incidents.

The logic of identifying an incident can be based on the movement of data points in the volume-occupancy plot for both upstream and downstream stations. Consider a volume-occupancy plot of a downstream station as shown in Figure 6.1. Under normal peak traffic operations, the data point will be in the region around point A. When an incident has happened, and after the last vehicle that passed the incident location at the onset of the incident has reached the downstream detector, the data point could shift to point B in the low-volume-low occupancy region. With the

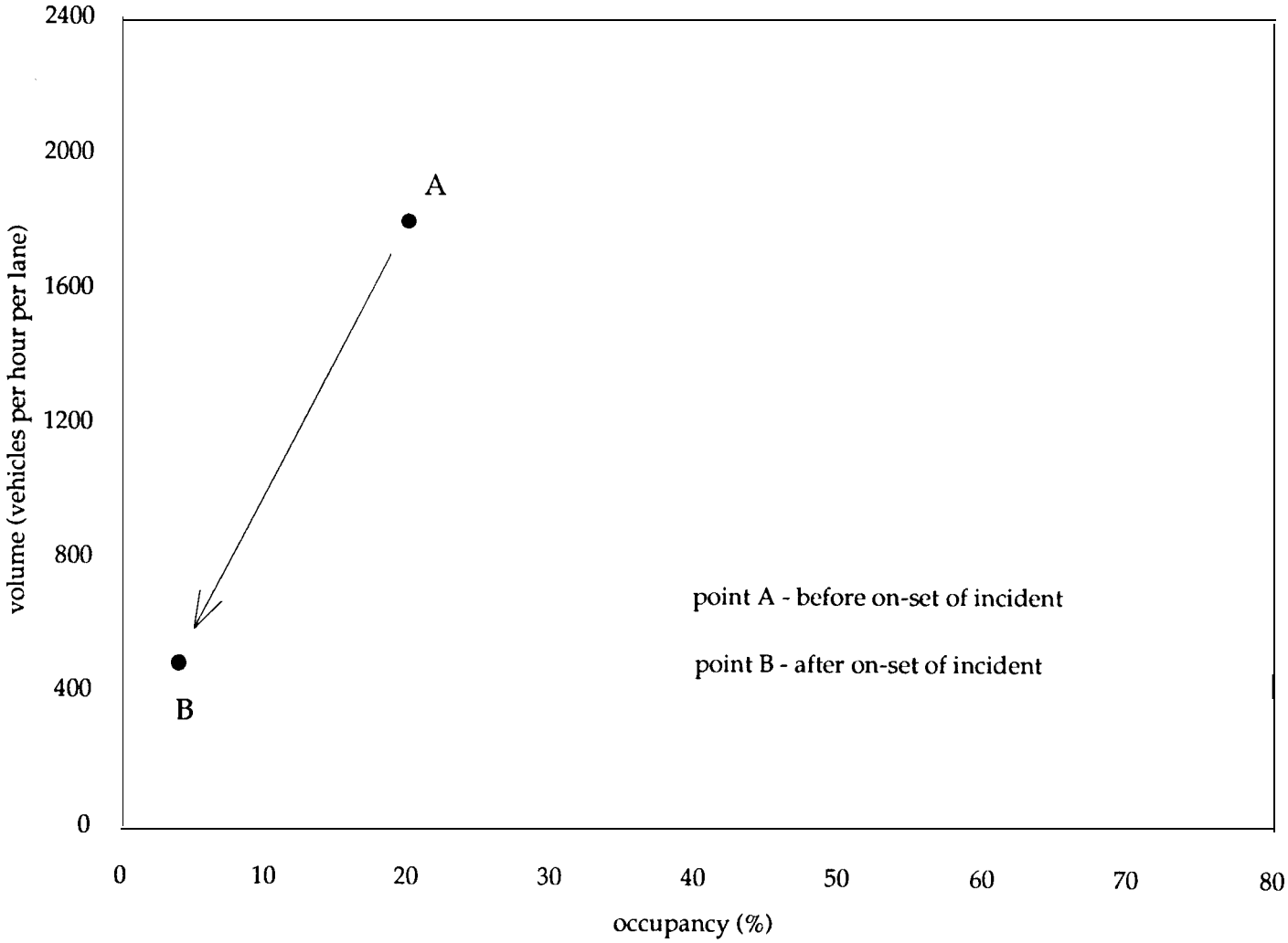

Figure 6.1 Movement of data point at downstream station at beginning of an incident

continuation of the incident, the downstream occupancy and volume will remain in the region.

On the other hand, the upstream data point, as illustrated in Figure 6.2, will remain in the region of normal traffic operation for a short period and then move to the region with higher occupancy values when the incident queue has reached the upstream station.

Consider the movement of flow conditions from point A to point B in Figure 6.1 for a station, which might be used to identify the on-set of incidents. One could calibrate, based on historical data, a boundary separating the regions for data points before and after incidents, such as the technique used in the McMaster algorithm (Persaud et *al,* 1990). Alternatively, the magnitude of movement of the data point, such as the relative measure of DOCCTD in the California algorithm, may be used to detect an incident. However, the boundary between incident and incident-free data points, and the magnitude of movement of the data point at the beginning of an incident depends on the prevailing conditions (such as volume and weather), and the severity of the incident. In the case of the neural network approach to incident detection, the associated patterns at the beginning of incidents under different conditions are recognized and distinguished, without an explicitly defined algorithm.

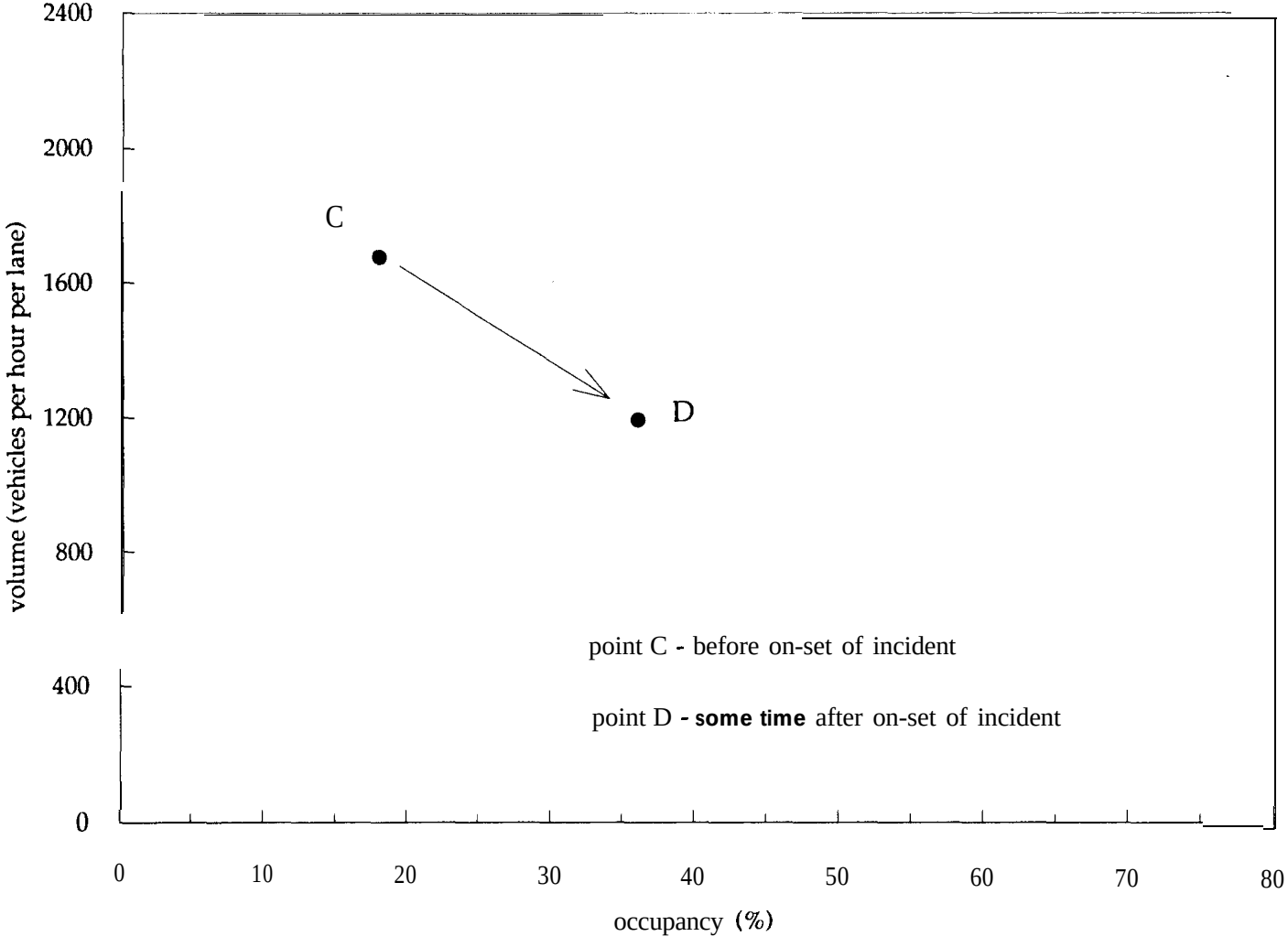

Figure 6.2 Movement of data point at upstream station at beginning of an incident

6.3 PROPOSED INPUT FEATURES

The simplest input pattern to the neural network model would be the downstream volume and occupancy values, averaged at 30 second intervals across all lanes, To detect if there is a sufficient drop in the volume and occupancy values, and the absolute values before and after the movement (such as from point A at t-l to point B at t in Figure 6.1), four input features, each feeding to an input PE would be sufficient. A data point at t-2 could be added to provide an indication to the neural net of the traffic flow condition prior to an incident. This would also help to eliminate false alarms due to random fluctuations in traffic flow. Inspection of data for less severe incidents at relatively low volume conditions has shown that the reduction in occupancy and volume at the beginning of the incidents may require three intervals to complete the movement. Therefore, the volume and occupancy values at the 3 intervals, namely t-2, t-l and t, are used as the basic input to the proposed neural network models.

Further inspection of the data has also shown that when an incident has occurred upstream of the test section, both upstream and downstream stations of the test section would experience the same pattern in the change of volume and occupancy, with the downstream station lagging the upstream station. At least one interval of data from the upstream station is therefore required to distinguish upstream incidents from incidents within the section. Under upstream incident conditions, the upstream station would have its data point in the low-volume-low-

occupancy region. On the contrary, for incidents within the test section, the upstream data point will remain out of the low-volume-low-occupancy region.

The MLF must also be designed to handle downstream incidents and shock waves coming from downstream. When there is an incident downstream of the test section and the queue has reached the downstream station of the test section, the data point will move to the region with high occupancy instead of approaching the origin of the volume-occupancy plot. In the case of a shock wave due to the removal of an incident downstream of the test section, when the wave front is crossing the downstream station into the test section, the data point in the volume occupancy plot would also move towards the origin. However, the magnitude of change would most likely be relatively small, or the change would be gradual, compared to a 'real' incident condition. In addition, the downstream of the wave front normally has higher flow rate (because there is no physical reduction of capacity).

With the above considerations, the first proposed neural network models would have the following input features:

upstream volume at t

upstream occupancy at t

downstream volume at t

downstream occupancy at t

downstream volume at t-l

downstream occupancy at t-l

downstream volume at t-2

downstream occupancy at t-2

This MLF's has 8 input PE's, one for each feature, and is referred to as the 2-station-3-interval net.

Several modifications to the above model have also been tested. Two input features at t-2 have been removed to form a 2-station-2-interval-net #1. This would provide a test of significance of the inputs at t-2 which account for the traffic conditions prior to incidents. This would help to reduce the time of computation and increase the ease of data management, in view of the thousands of detector stations in the entire freeway system in the greater Los Angeles area. Another modification is the removal of two input features from t-l. This neural net is called 2-station-2-interval-net #2. The reason for avoiding data from interval t-l is that for some incidents, the subsequent movement of data points at the beginning of the incidents is relatively small. Using only data from t-2 and t would make the shift more obvious.

Two other neural net models, using data from a single station, have also been considered. The first single station model, named downstream-

station 3-interval net, is the same as the 2-station-3-interval-net without the upstream input. This single station model is expected to detect incidents upstream of the test section, in addition to the incidents within the test section. If this model proved to be superior in detecting incidents within the test section, a control mechanism would have to be implemented so that when an incident is detected upstream of the section, the incident warning at sections located downstream would be turned off or ignored.

To compare the difference in time-to-detection when using upstream data, a second single station neural net was also tested. This model, named upstream-station 3-interval net had six input features all from the upstream station of a test section. The input features were the upstream occupancy and volume, at t-2, t-l and t. The input features of the five neural network models are summarized in the following Table 6.1.

6.4 OUTPUT FEATURES

Since the nets are designed to detect the movement of data points in a volume-occupancy plot, only two states are of interest: beginning of incident or all other conditions. Thus these two output states can be represented by 1 PE in the output layer. The output PE will have a desired value of 1 for state 2 (beginning of incidents) and 0 for state 1 (other conditions).

Table 6.1 Input features of the proposed neural nets

station location	time inter -val	volume or occupancy	2-station 3-interval net	2-station 2-interval net #1	2-station 2-interval net #2	upstream-station 3-interval net	downstream -station 3-interval net
upstream	t-2	volume					+
		occupancy					+
	t-1	volume					+
		occupancy					+
	t	volume	+	+	+		+
		occupancy	+	+	+		+
down-stream	t-2	volume	+		+	+	
		occupancy	+		+	+	
	t-1	volume	+	+		+	
		occupancy	+	+		+	
	t	volume	+	+	+	+	
		occupancy	+	+	+	+	

The movement of detector data points at the beginning of incidents as depicted in Figure 6.1 is to be detected by the neural nets. Such a pattern is to be distinguished from other traffic events. Since detecting the on-set of incidents is of primary interest, input patterns to the MLF's were to be classified into 2 states: state 1 and 2. State 2 represents the beginning of an incident and state 1 other conditions. Only one PE in the output layer is required to represent these two states. In determining the output, a value equal to or greater than 0.5 represents state 2 while values of less then 0.5 are state 1.

6.5 CONCLUDING COMMENTS

A total of five MLF models were proposed in this study. These five models used different combinations of station average volume and occupancy from the upstream and downstream stations, of up to three intervals, as inputs. The number of PE's in the input layer ranged from to 12 to 16. All the models have one PE in the output layer, which classified the traffic conditions in a freeway section into one of two states, either as the beginning of incidents or other conditions.

CHAPTER 7

TRAINING DATA

7.1 INTRODUCTION

The proposed neural net models were trained with data for incident and non-incident conditions. However, field detector data with detailed information about incidents is difficult to obtain. An extensive set of freeway surveillance data has been documented by the Technological Service Corporation (Payne and Helfenbein, 1976). This set of data consists of 121 recorded incidents at three freeways in Los Angeles and is the data set used by Payne et al (1976) in the the calibration of the California algorithms. The National Technical Information Services (NTIS) could not provide these archived data because a substantial portion of it has apparently been destroyed. Subsequent enquiries with researchers and the Federal Highway Adminstration (FHWA), who have experience in using the data set, revealed that part of the data set has been lost. Furthermore, all the recorded incidents are not at the same location. As the freeway geometry and traffic demand vary at different locations, any incident detection algorithm or model has to be calibrated or trained with local data. Such has been the practice in the implementation of the California algorithm in Los Angeles freeways (Arceneaux et al, 1989).

We therefore decided to use detector data generated from a computer simulation program to train the neural net models. The advantages of using simulation data are two-fold: (1) it allows the generation of many types of incidents under different flow conditions for which actual data would be extremely difficult to obtain; and (2) the duration, location and nature of an incident are deterministic, enabling assignment of the desired states of the output vectors for training and evaluation of the model.

7.2 SIMULATION DESIGN

Since detector data is required for every 30 second interval, a microscopic freeway simulation model capable of counting vehicle volume and occupancy at each loop detector had to be used. INTRAS was thus selected for the simulation of incidents. INTRAS permits a microscopic simulation of freeway operation under incident conditions, and outputs detector occupancies and volumes as needed for incident detection purposes. INTRAS has been extensively calibrated and validated to represent general freeway traffic operations in Southern California (Goldblatt, 1980; Skabardonis et *al*, 1989). The most recent version of INTRAS was obtained from FHWA and made operational for purposes of this study.

7.2.1 The Test Section

As the neural net models were to be trained with localized data, the study concentrated on a freeway section with typical geometry. Detector data for this freeway section during incident and incident-free conditions were generated for training and testing of the MLF's.

Among the freeway sections in both directions of the SR-91 testbed, the westbound section between Euclid St and Brookhurst St was selected because it had the most reported incidents in a recent sampling of the California Highway Patrol (CHP) records. Actual detector data during incidents may be readily available for future testing, due to the frequent occurrence of incidents. In addition, this section has geometry and detector configurations that are typical in the testbed. A schematic diagram of the test section, is shown in Figure 7.1. The section length, between detector stations 7 and 8, is 0.972 miles. There are 3 through lanes, with an on-ramp from Euclid St and an off-ramp to Brookhurst St between the two stations. An auxiliary lane connects the on-ramp and the off-ramp. All the loop detectors are installed at through lanes in upstream of the on-ramps.

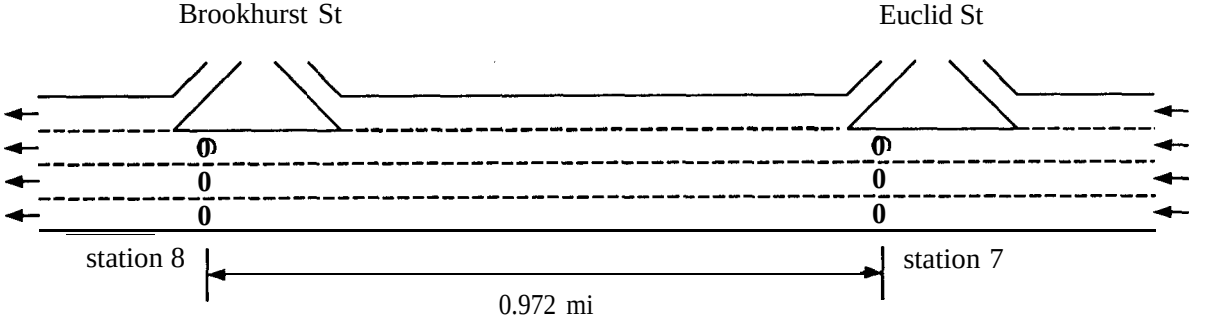

Figure 7.1 Schematic diagram of the test section

7.2.2 Traffic Volume

The MLF's were trained to detect incidents under a variety of traffic flow conditions. The only available traffic volumes, collected on June 8 1987 by Caltrans were examined to establish several flow levels. After discarding intervals in which detectors had malfunctions, cumulative distribution of 15 minute volumes at station 7 from 7.00 a.m. to 7.00 p.m. is plotted in Figure 7.2 (since this station counts the flow of vehicles entering the test section). From the distribution and range of volumes, it was decided to perform simulations at three different flow levels, respectively. The minimum flow is about 1300 vphpl, while the 50th percentile and maximum flows are 1800 and 2200 vphpl, respectively. Training the neural network with a large range of volume helps improve its robustness in detecting incidents under a variety of flow conditions.

7.2.3 Location of Incidents

For the neural network models to learn to detect incidents that occur anywhere within the test section, incidents were generated with uniform spatial distribution along the section length. For simulation runs, incidents within the test section were located at five locations. These were: 800, 1600, 2400 and 3200 ft downstream of node 14, and 600 ft downstream of node 15, as indicated in Figure 7.3.

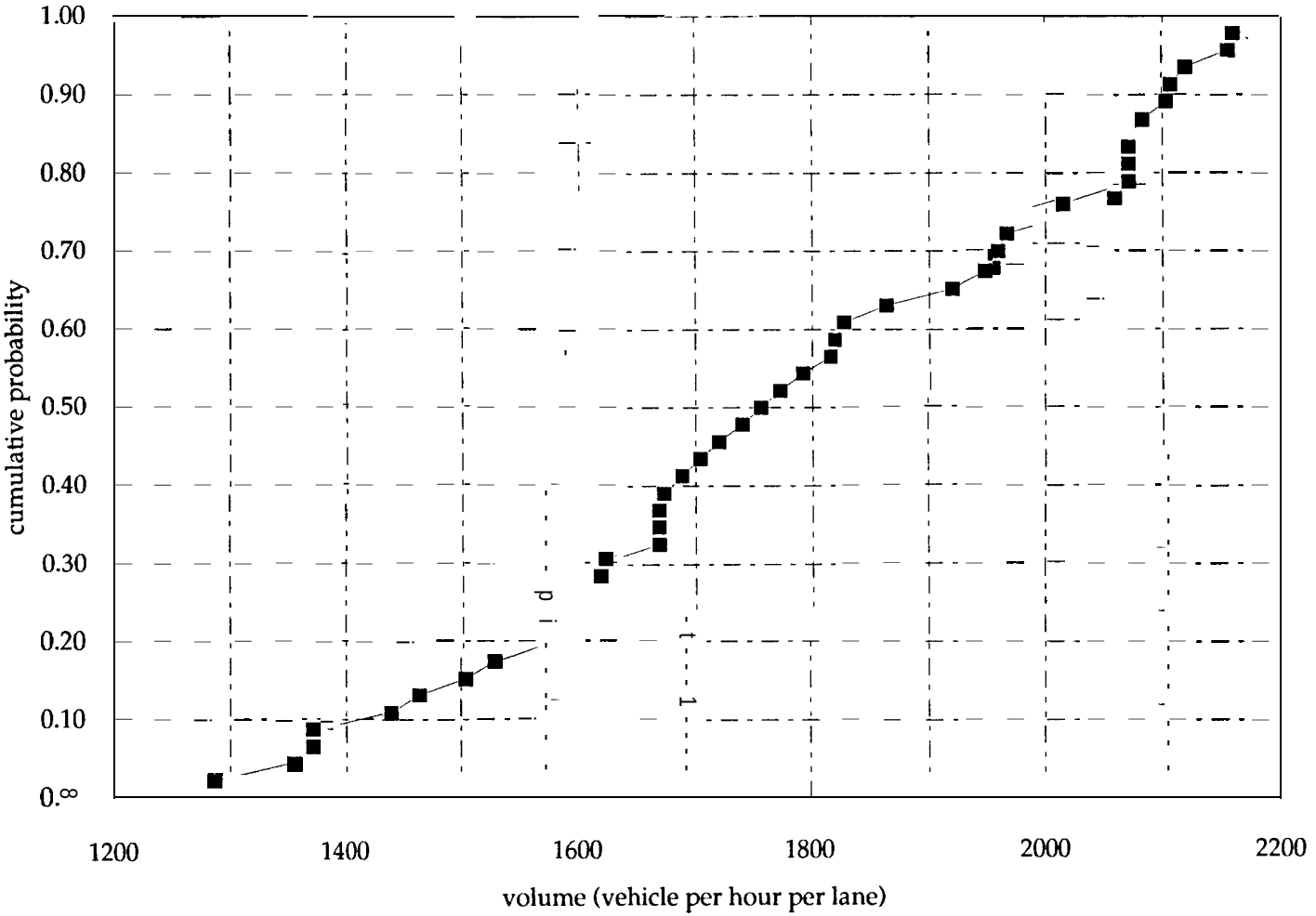

Figure 7.2 Cumulative probability distribution of 15 minute day time volume at station 7 on June 8 1987

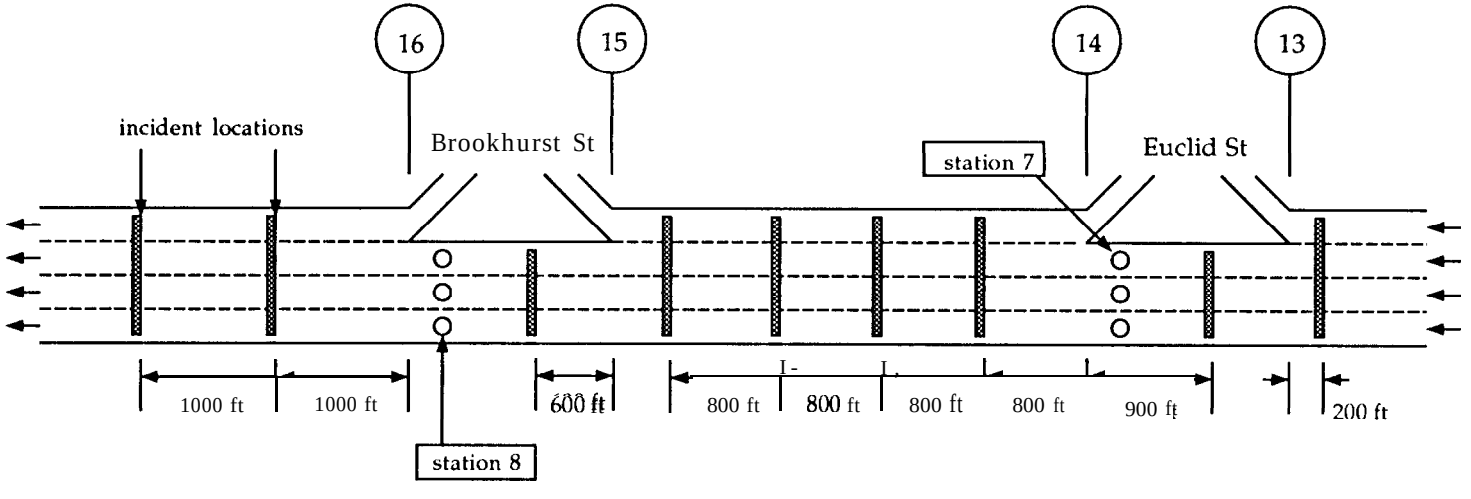

Figure 7.3 Incident locations and INTRAS node numbers

As already discussed in sections 6.2 and 6.3, and based on experience with the simulation data, incidents that occur upstream or downstream of the test section may trigger false alarms. Incidents have to be simulated upstream and downstream of the test section so that the MLF's can be trained to recognize these false alarms (incidents not occuring within the test section). Two locations, 900 ft upstream of node 14 and 200 ft upstream of node 13 were identified as the upstream incident locations. Another 2 incidents locations, 1000 and 2000 ft downstream of node 16 were also identified for this purpose. These locations are indicated in Figure 7.3.

7.2.4 Patterns of Lane Blockages

At each incident post mile, all the possible lane-blockage patterns were identified and simulated separately as individual incidents. The severity of an incident was accounted for by the number of lane blocked. Incidents that involve vehicles parked on the shoulder were not considered because they do not cause a physical blockage of the main lanes. Inspection of INTRAS detector output showed that such incidents did not result in significant changes in detector data. They normally would also result in smaller delay.

In locations where there are three travel lanes, l-lane blockage, 2-lane blockage and 3-lane blockage incidents are possible. For incidents with multiple lane blockages, all the affected lanes are placed adjacent to

each other. Hence, for a location with four lanes, there are 10 incident patterns: four l-lane blockage incidents (1 for each lane), 3 2-lane blockage incidents, two 3-lane blockage incidents and one for complete blockage. At sections where there are 3 lanes, there are six different configurations.

With five incident locations in the test section and all configurations at each location, there would be 46 possible incidents for any given volume, Correspondingly, there are 20 simulations for upstream incidents and 16 simulations for downstream incidents at any given volume. Taking the three flow levels into account, there are a total of 138 incidents within the test section and 108 incidents at upstream and downstream sections.

7.3 GENERATION OF DETECTOR DATA WITH INTRAS

The 5-mile testbed in Figure 7.4, including on and off ramps and the exact locations of the detectors, was coded into nodes and links in an INTRAS input file. Exit signs were placed approximately 1 mile upstream of the exits. Acceleration and deceleration lanes, if present, were assumed to be 300 ft in length. A zero grade was assumed for the freeway. All the detectors used in the testbed were single loops. The effective length of each loop was set to 4 ft. Other important parameters are listed below:

free flow speed on freeway links = 65 mph

free flow speed on ramps and surface links = 55 mph

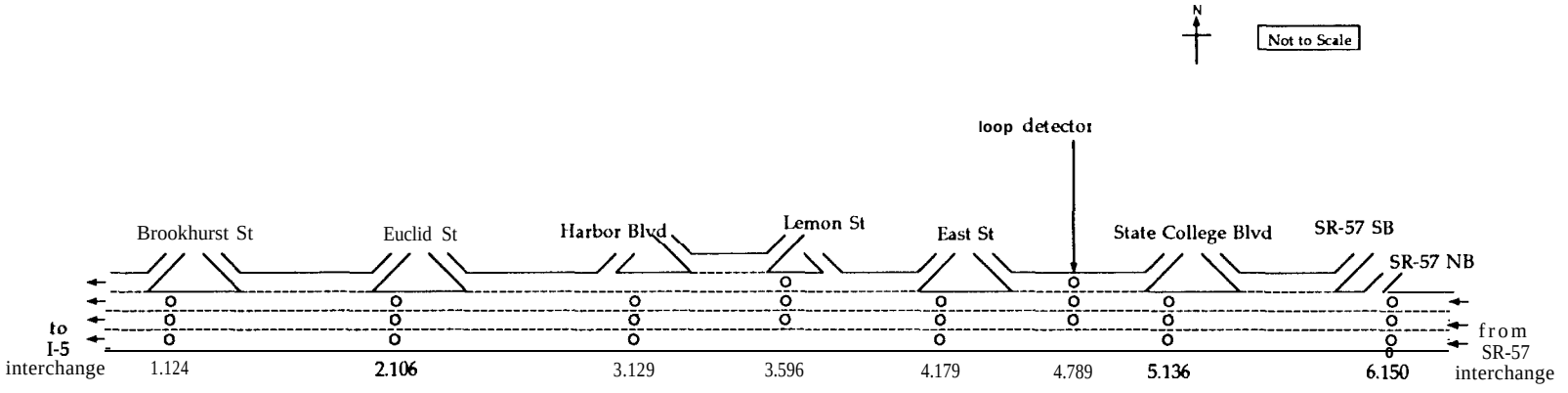

Figure 7.4 Schematic diagram of SR-91 Freeway testbed (westbound)

rubber necking factor = 40%

average vehicle length = 15 ft

vehicle composition:

low performance passenger cars	= 42%
high performance passenger cars	= 50%
buses	= 0%
single unit truck	= 0%
truck trailers	= 8%

car-following parameters = default values

The flow levels at which simulations were carried out were taken from the three 15 minute intervals selected in Section 6.2.2. Since the available Caltrans data did not include ramp volumes, input volumes to INTRAS at on-ramps were estimated from the conservation of vehicles, using volumes at various mainline detector stations. When the law of conversation could not be applied (due to the absence of on-ramp detector at upstream end of the section and off-ramp detector at downstream end of the section), on-ramp volumes of 100 vph were assumed for low flow level, and 300 vph was used for simulations at moderate and high flow levels. The exit volumes were then deduced from the law of conservation.

To reduce the data storage requirements and simulation run time, each simulation was performed in INTRAS for 35 minutes (excluding startup time). In each run, the system was allowed five minutes of startup time to reach equilibrium, after which detector data were collected for 35

minutes. After the startup time, normal freeway operation was allowed for five minutes for the collection of data in incident-free conditions. The designed incident was placed from 315 to 615 seconds into the run. The 5-minute incident duration was adequate for any queue to spill back to the upstream station. Another 25 minutes of simulation was necessary for the recovery stage, letting the traffic restore to normal operating conditions. The duration of the simulation was sufficient to ensure the restoration of normal traffic operation under the most severe incident at high volume. Therefore, for each incident, 35 minutes of detector data at stations 7 and 8 were collected at 30 second intervals.

A total of 246 incident simulation runs were performed with INTRAS. In each run, traffic operations for the entire testbed, in the westbound direction as shown in Figure 7.4, were simulated, and detector data were extracted from station 7 and 8 every 30 seconds. Simulation of the entire testbed operation was necessary to prevent queues spilling beyond the entry node of the network so that recovery of traffic operations after incidents could be monitored. The extracted data were in the form of 30 second average volume and occupancy.

Detector data from 246 INTRAS simulation runs were used to form two data sets: the training data set (for training of neural nets) and test data set (for independent test). The simulation cases were randomly assigned to the two data sets such that each data set consists of 123 mutually exclusive simulation runs, of which 69 of them are incidents within the test section while the remaining are incidents at upstream and downstream of the test section. Both data sets have almost an equal

number of incidents in terms of severity (lane blockage patterns) and location of occurrence. With 123 simulation cases, there are 8364 vectors in each data set. The content of the two data sets are summarized in Table 7.1.

Table 7.1 Number of incidents in training and test data sets

	training set	test set
no. of incidents within test section	69	69
no. of upstream incidents	24	24
no. downstream incidents	30	30
total no. of incidents	123	123

7.4 ASSIGNMENT OF DESIRED STATES

From each simulation run, outputs of individual loop detectors, in the form of 30 second cumulative volume and occupancy, were extracted from the output files. At each 30 second interval, data collected from individual loop detectors were aggregated to form station average values (across all lanes). Because of the design of the input features of the neural net models, which require up to three intervals of detector data from a station, only 68 input vectors can be extracted from 35 minutes of data in each simulation run.

For each simulation run, the desired output state of each input vector was assigned by manually inspecting the station's average volume and occupancy with time. For each of the 138 incidents that occur within the test section, only 1 vector is assigned state 2 and others are assigned state 1. For incidents which occurred outside of the test section, all the vectors had desired output state of 1.

Because all the incidents were simulated to begin at 315 seconds into the run time, the movement of the downstream station's data point from incident-free region to the low-volume-low occupancy region at the downstream station should have been completed by 330, 360 or the latest 420 seconds. Thus for training and testing of the 2-station-3-interval net, 2-station 2-interval net #1 and the downstream-station 3-interval net, a desired state of 2 was assigned for vectors within this time frame, based on

the description of the data pattern in Sections 6.2 and 6.3. These three MLF models used the same set of desired output states.

Initially, the same desired states were also applied to the 2-station 2-interval net #2. However, the training results were not satisfactory because many vectors immediately after the desired state 2 vectors also had the same patterns in the input features. It was therefore decided to increase the number of desired state 2 vectors from one to two, for some incidents. The additional state 2 vectors were assigned immediately after the first original state 2 vectors, if necessary. Because of the assignment of two state 2 vectors to an incident in some cases, repeated warning (detection of state 2) in incident detection runs should not be treated as false alarms.

A separate set of desired states was assigned for the upstream-station 3-interval net. The state 2 vector in each incident within the test section was assigned to a vector when there was sufficient increase in upstream occupancy and reduction of upstream volume.

The entire process of assignment of desired states was performed by the same person. Due to the inherent variability in human judgement, some random error in the assignment of state 2 vectors was expected, even after checking the data several times. Such inconsistency can be detected by inspection of the classified states produced by the neural network models after some preliminary training. Normally the error in the assignment was within 1 interval. Once the states of the vectors were considered correctly assigned, the order in which the vectors appear in the

data set was randomized before the commencement of training. With 123 simulation cases, there were 8364 vectors in each of the training and test data sets. The number of state 1 and state 2 vector are summarized in Table 7.2.

Table 7.2 Number of vectors in training and test data sets

	training set	test set
total no. of state 1 vectors	8295*	8295*
total no. of state 2 vectors	69*	69*
total no. of vectors	8364	8364

* The 3-station 2-interval net #2 had 8254 state 1 vectors and 110 training vectors in the training data set and 8257 state 1 vectors and 107 state 2 vectors in the test data set

7.5 CONCLUDING COMMENTS

Detector data used in this study were generated from INTRAS, a microscopic freeway traffic simulation model. A total of 246 simulations runs were performed, and loop detector data at the upstream and downstream ends of the test section were extracted to form two data sets.

The training and test data sets were formed to train and test the MLF's respectively. Each data file consists of 123 incidents which has 8364 vectors. Each vectors were manually assigned an output state of either 2, which represents the beginning of incidents, and state 1, for other conditions.

CHAPTER 8

RESULTS OF NEURAL NET TRAINING

8.1 THE TRAINING PROCEDURE

The training of MLF's was performed with the backpropagation algorithm as outlined in Section 4.3. For each proposed model, the 'optimum' number of PE's in the hidden layer was determined as part of the training process. This number depends on patterns in the training vectors (i.e. the patterns presented to the input and output PE's). The training of each MLF was thus performed in two stages. The first stage was to select the optimum number of hidden PE's and the second stage was devoted to fine tune the performance of the optimum network.

In the first stage, 15 trainings sessions were performed, each with a different number of hidden PE's from two to 16. For each given number of PE's in the hidden layer, all the training vectors were presented to the net for 1000 cycles, using a learning rate η = 0.1, and a momentum rate a = 0.9. The initial weights were randomly assigned values between -1 and 1. At the end of 1000 cycles of training, the SSE and percentage correct recall of the training vectors were extracted to measure each net's performance with different number of hidden PE's. In addition, the trained networks were applied to the test data set. The percentage correct recall of the test vectors was also used in the evaluation. The network with the optimum

numbers of PE's in the hidden layer should ideally have a low SSE, and high percentage of correct recall of vectors for the two states in both the training and the test data sets.

After the optimum number of hidden PE's was selected for a proposed model, the second stage involved a continuation of training of the selected net to improve the accuracy in classification. Generally, training was performed up to 2000 cycles. The best performance at either 1000 or 2000 cycles of training was taken as the final version of the trained network and was used for off-line incident detection evaluation.

8.2 RESULTS OF TWO-STATION THREE-INTERVAL NET

The 2-station 3-interval net has sixteen input features. By varying the number of hidden PE's from two to 16, 15 different network configurations were generated and each trained for 1000 cycles. The SSE of these networks at 1000 cycles is plotted in Figure 8.1. For seven, eight, nine and 16 hidden PE's, the SSE was been reduced to almost 0. This indicates that the network parameters have converged to classify the training vectors.

The trained parameters at 1000 cycles, for the different numbers of hidden PE's, were applied to classify the training and test vectors. Since state 2, the beginning of an incidents, is of primary interest, the percent recall of the 69 state 2 vectors in each data set was plotted in Figure 8.2.

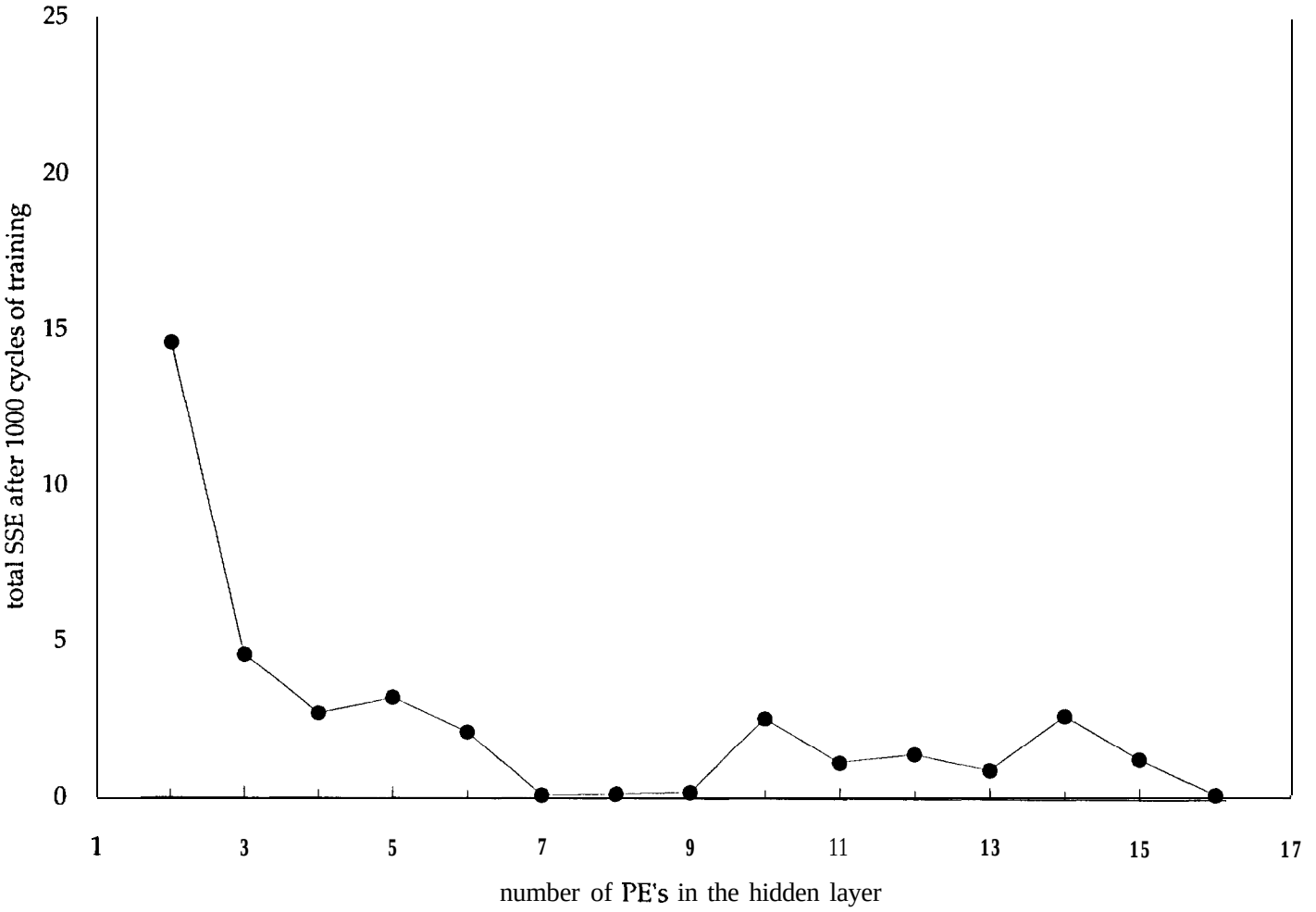

Figure 8.1 SSE for the 2-station 3-interval net with different numbers of hidden PE's

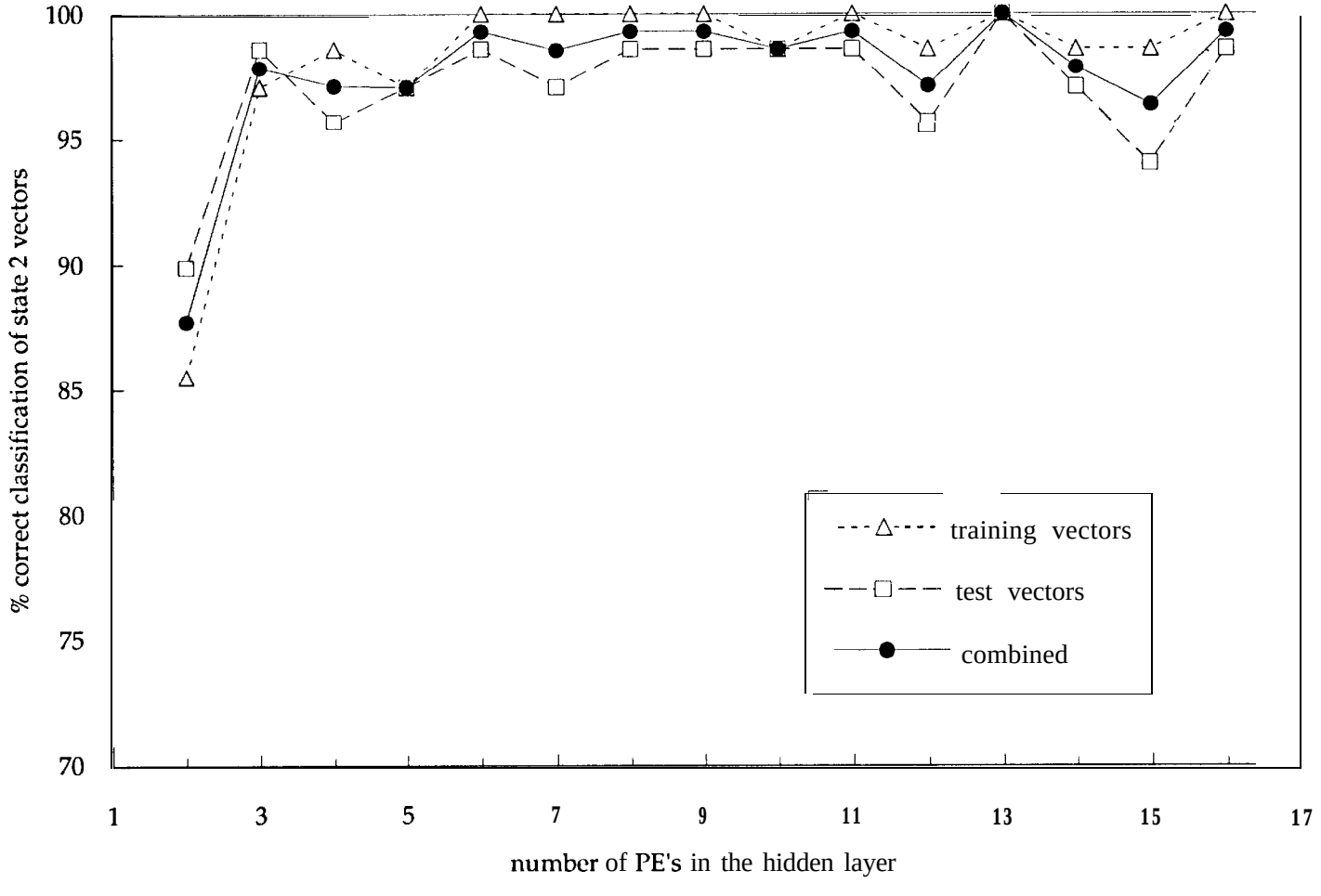

Figure 8.2 Percent correct classification of state 2 vectors for the 2-station 3-interval net with different numbers of hidden PE's

Generally, the networks had slightly better rate of recall for the training vectors than the test vectors. This was expected as the network parameters were adjusted to 'fit' the training data. Taking the two data sets into account, the combined rate of recall of state 2 vectors gives an indication of the detection rate of the model. The best rate occurred for the network with 13 hidden PE's, which had 100% correct recall of all the state 2 vectors. This implies that 100% detection of incidents was achieved.

Since there are 2 states in the neural net's output, the false alarm rate can be infered from the number of state 1 vectors being misclassified as state 2 vectors. Because there are 8295 state 1 vectors in each data set, the number of misclassificcation is of a very small percentage and only the absolute number is taken used for comparison. The total number of misclassified state 1 vectors from the two data sets is plotted in Figure 8.3. The network with 13 hidden PE's had the lowest number, with only 1 misclassified state 1 vector. Thus, 13 hidden PE's was chosen as the optimum number for the 2-station 3-interval net.

The optimum network configuration of 13 hidden PE's was trained up to 2000 cycles. The SSE was reduced from 0.111 at 1000 cycles to 0.039 at 2000 cycles. However, the test data set had one misclassified state 2 vector and one additional misclassified state 1 vector. It may be that the network parameters adjusted themselves too much to memorize the training vectors but lost generality in classifying similar patterns which they had not yet encountered. In such a case, the network is said to be 'over trained'. The best training results were still with the net that had 13

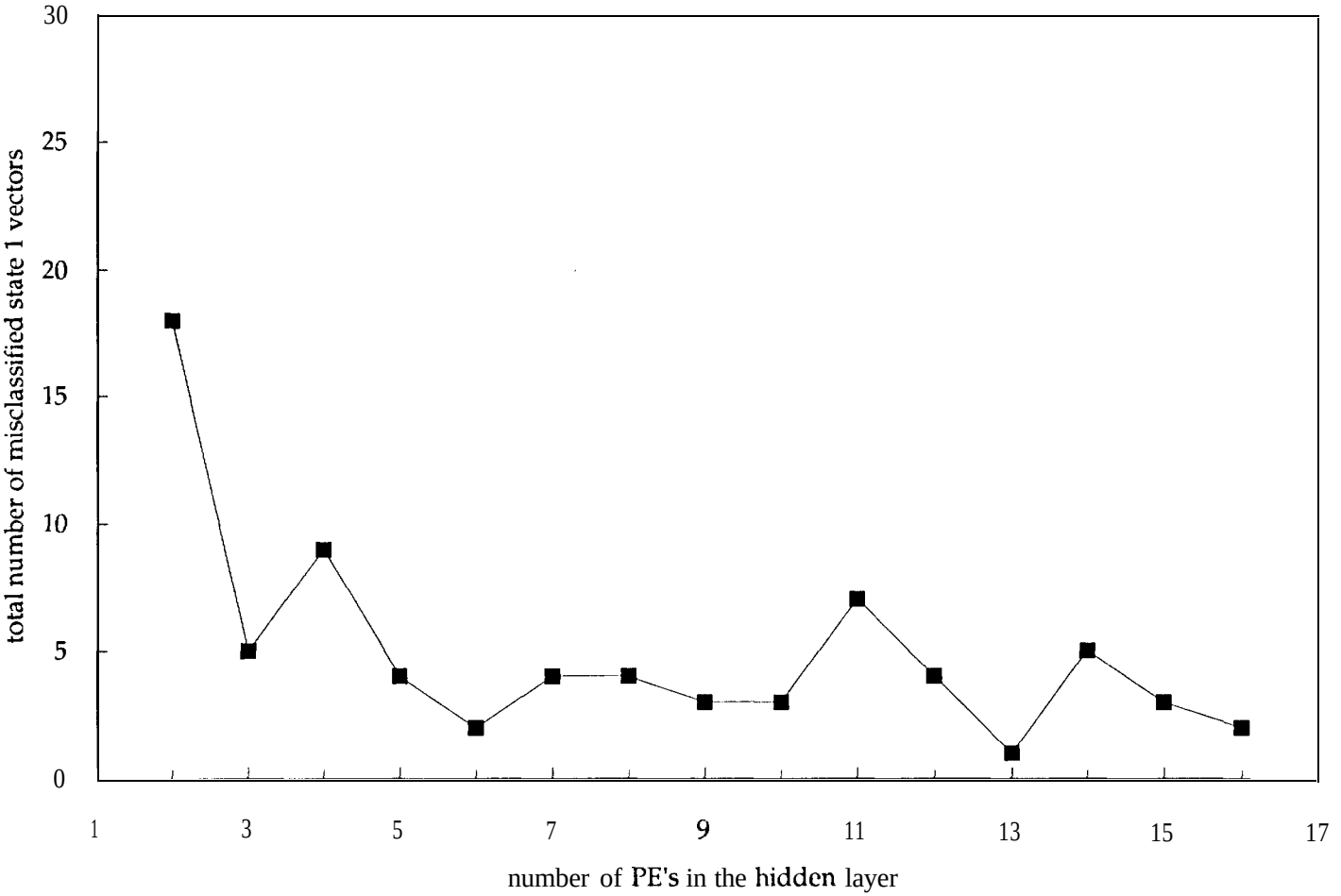

Figure 8.3 Total number of misclassified state 1 vectors for the 2-station 3-interval net with different numbers of hidden PE's

hidden PE's, at 1000 training cycles. The associated network parameters were subsequently used for off-line incident detection run.

8.3 RESULTS OF TWO-STATION TWO-INTERVAL NET #1

A similar training technique was applied to the Z-station Z-interval net #1 to select the optimum number of hidden PE's. From the plot of SSE at 1000 cycles of training for different numbers of hidden PE's in Figure 8.4, the networks with 9 and 11 hidden PE's have converged the most. However, when the percent correct classification of state 2 vectors is examined in Figure 8.5, the networks with 11 and 14 hidden PE's have higher recall rates. These two nets have a combined correct classification rate of 98.6%. They classified all the state 2 vectors in the training set correctly but missed two vectors in the test data set. The plot of total number of misclassified state 1 vectors in Figure 8.6 suggests that 10 hidden PE's gives optimum results. However, as the neural nets are designed to detect incidents, the ability to correctly detect state 2 inputs is the most important performance criterion. The networks with 11 and 14 hidden PE's both have the highest classification rate for state 2. But the one with 14 hidden PE's has five misclassified state 1 vectors compared to eight for the net with 11 hidden PE's. The decision was therefore made to continue to train the nets with 11 and 14 hidden PE's to 2000 cycles.

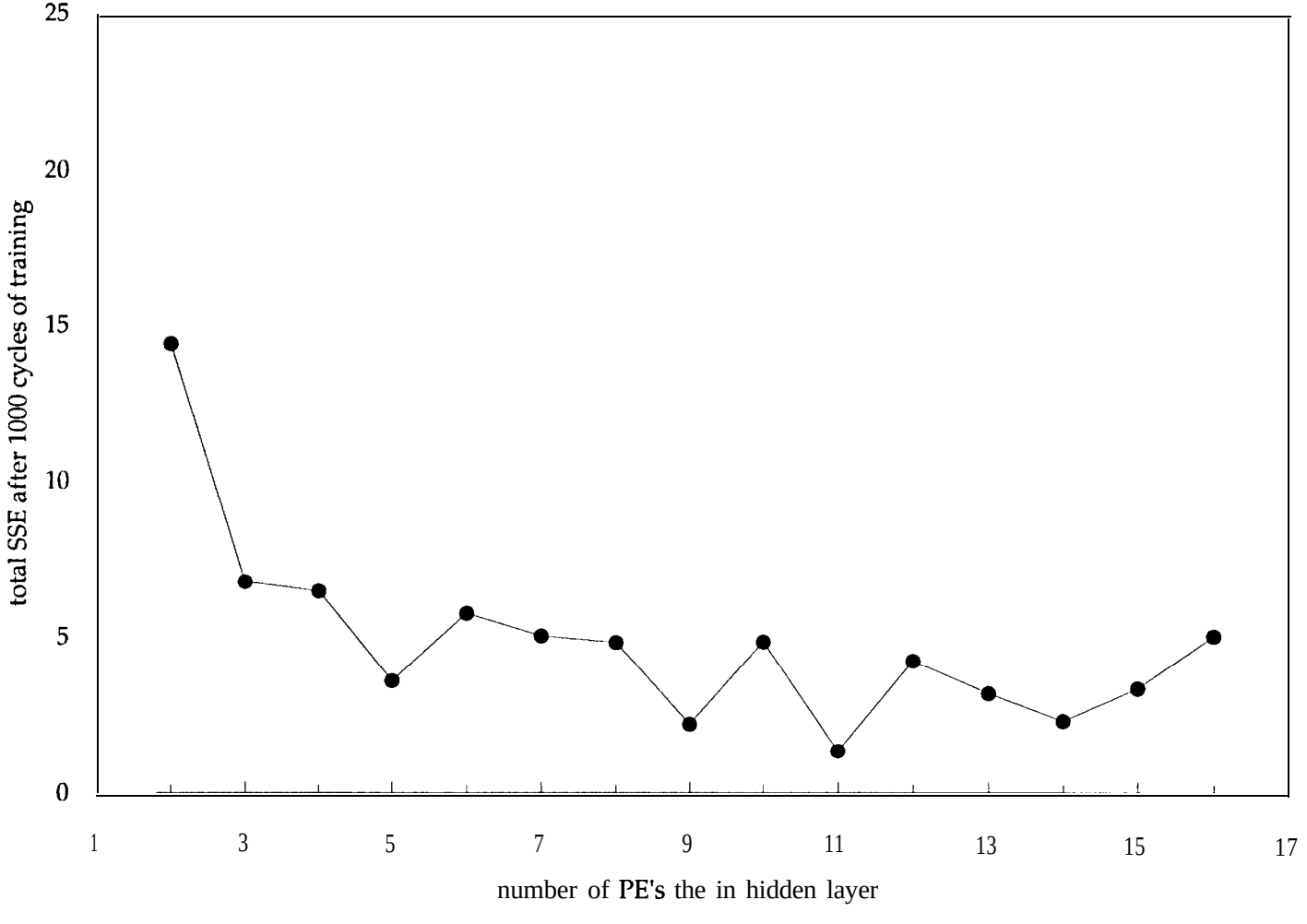

Figure 8.4 SSE for the Z-station Z-interval net #1 with different numbers of hidden PE's

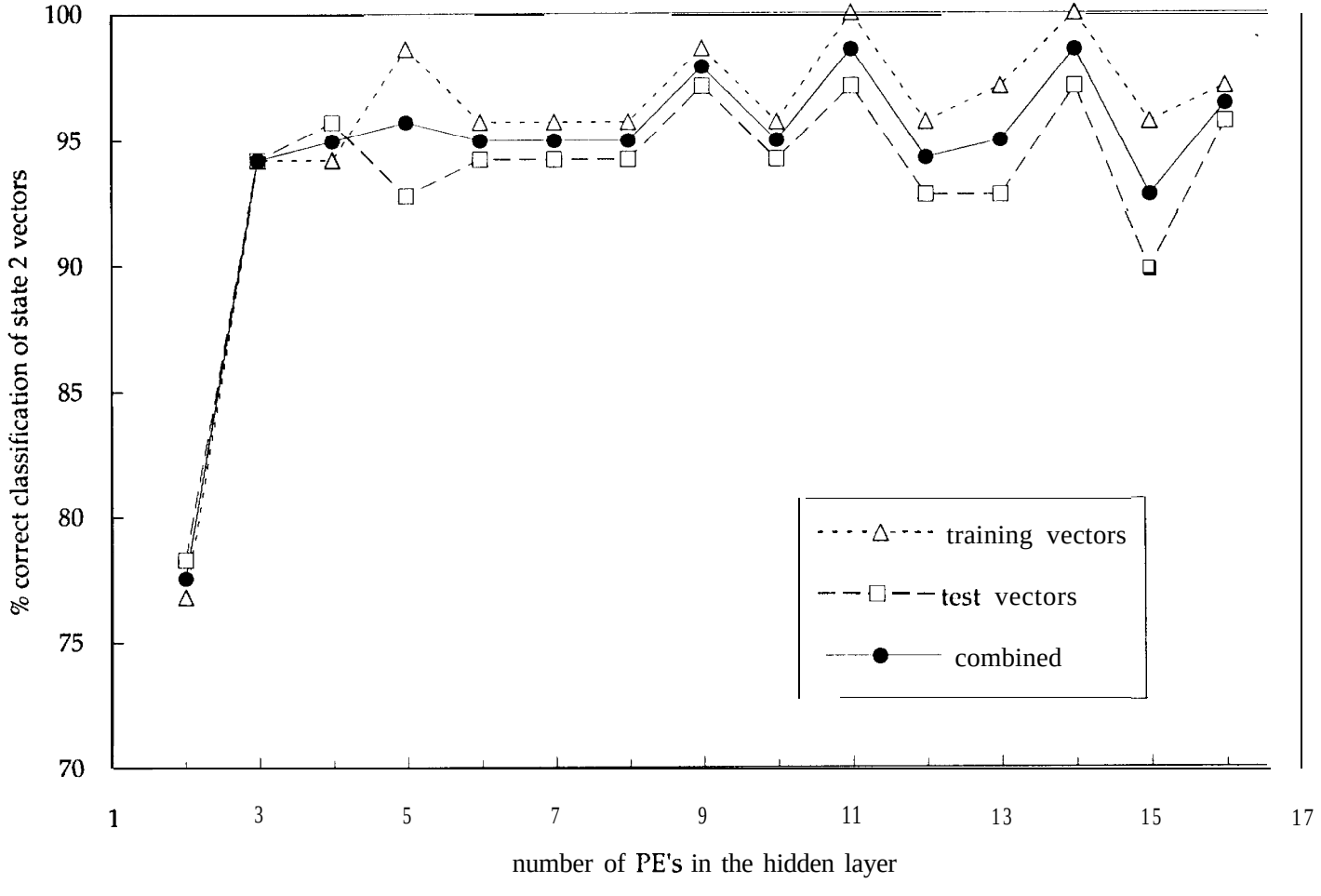

Figure 8.5 Percent correct classification of state 2 vectors for the 2-station 2-interval net #1 with different numbers of hidden PE's

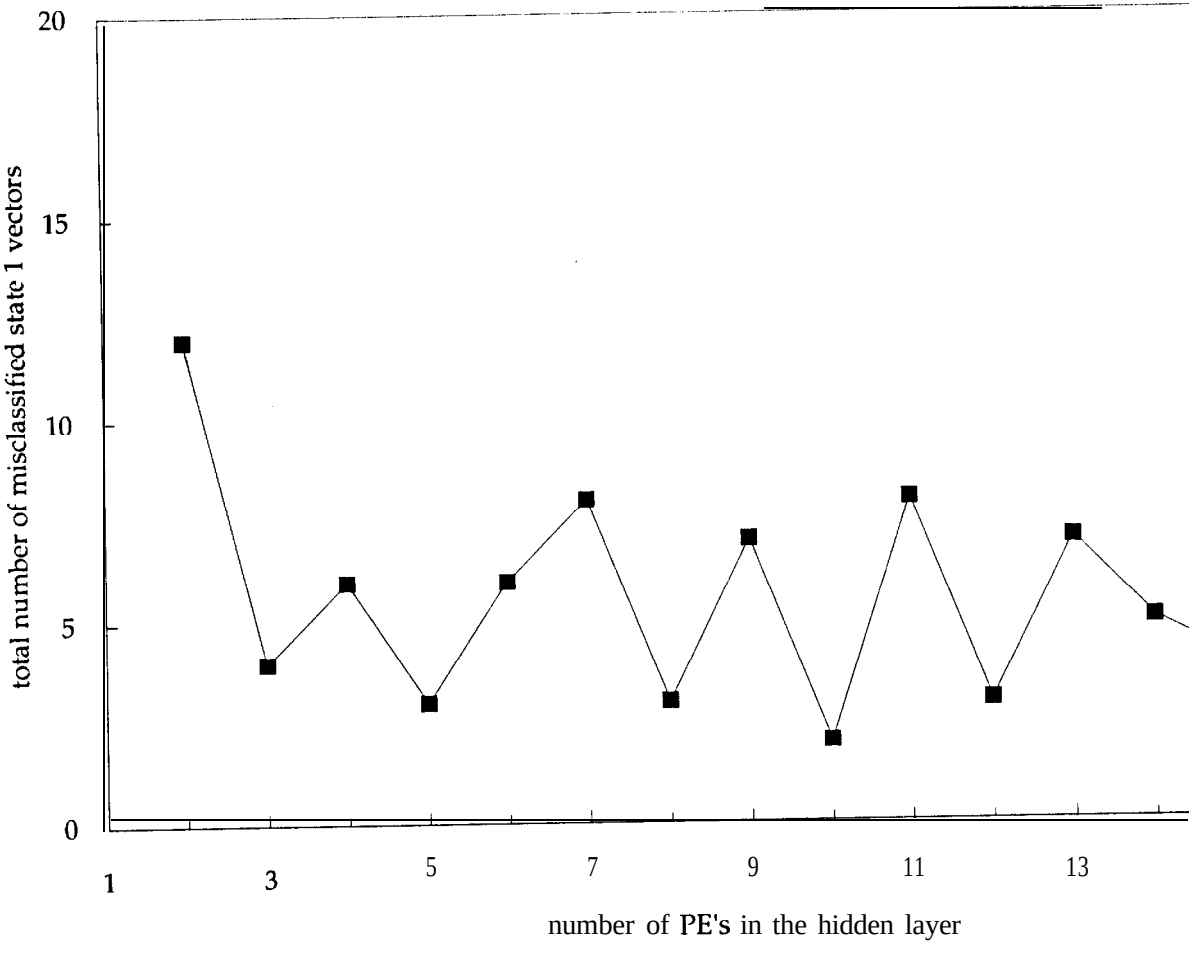

Figure 8.6 Total number of misclassified state 1 vectors for the Z-station 2-interval ne
numbern of hidden PE's

The results for the 2-station 2-interval net #1 at 1000 and 2000 cycles of training are summarized in Table 8.1. For both network configurations, the SSE was reduced by increasing the cycles of training. However, the reduction of SSE did not lead to improvement in the rate of recall of state 2 vectors. The only improvement with continued training was the reduction of the number misclassified state 1 vectors for the net with 11 hidden PE's. From Table 8.1, the best result was obtained with 14 hidden PE's at 1000 cycles of training. This network gaves the highest rate of recall of state 2 vectors and the least number of misclassified state 1 vectors.

Table 8.1 Summary of training results of 2-station 2-interval-nets #1

no. of hidden PE's	no. of training cycles	SSE	% correct recall of state 2 vectors on both data sets	# of misclassified state 1 vectors
11	1000	1.345	98.6	8
11	2000	1.088	98.6	7
14	1000	2.308	98.6	5
14	2000	2.062	97.1	5

8.4 RESULTS OF TWO-STATION TWO-INTERVAL NET #2

The results of the first stage of training of the 2-station 2-interval nets #2 are shown in Figures 8.7, 8.8 and 8.9. From Figures 8.8 and 8.9, it can be deduced that this network design needs at least four hidden PE's to achieve satisfactory performance. The SSE, for four or more hidden PE's, falls in a narrow band between 5.0 and 8.0. Similarly, for six or more hidden PE's, the combined correct classification rate for state 2 vectors is above 98%. The net with seven hidden PE gave the highest combined classification rate of 99.5%. The error resulted from a misclassification of a state 2 vector in the training data set (as a departure from other proposed network models as explained in Section 7.4, this model had 110 and 107 state 2 vectors in the training and the data sets respectively). This network, at 1000 cycles of training, resulted in 18 misclassifications for state 1 vectors. The number of misclassifications of state 1 vectors, as plotted in Figure 8.9, was in the range of 8 to 20. The network with seven hidden PE's had the second highest number of misclassifications. However, as we have given priority to the accurate classification of state 2 vectors, this number of hidden PE was selected as the optimum design despite the relatively high error rate for state 1. The 2-station 2-interval net #2 with 7 PE's was trained up to 2000 cycles. Based on the results of the the net at 1000 and 2000 cycles, a final selection was then made.

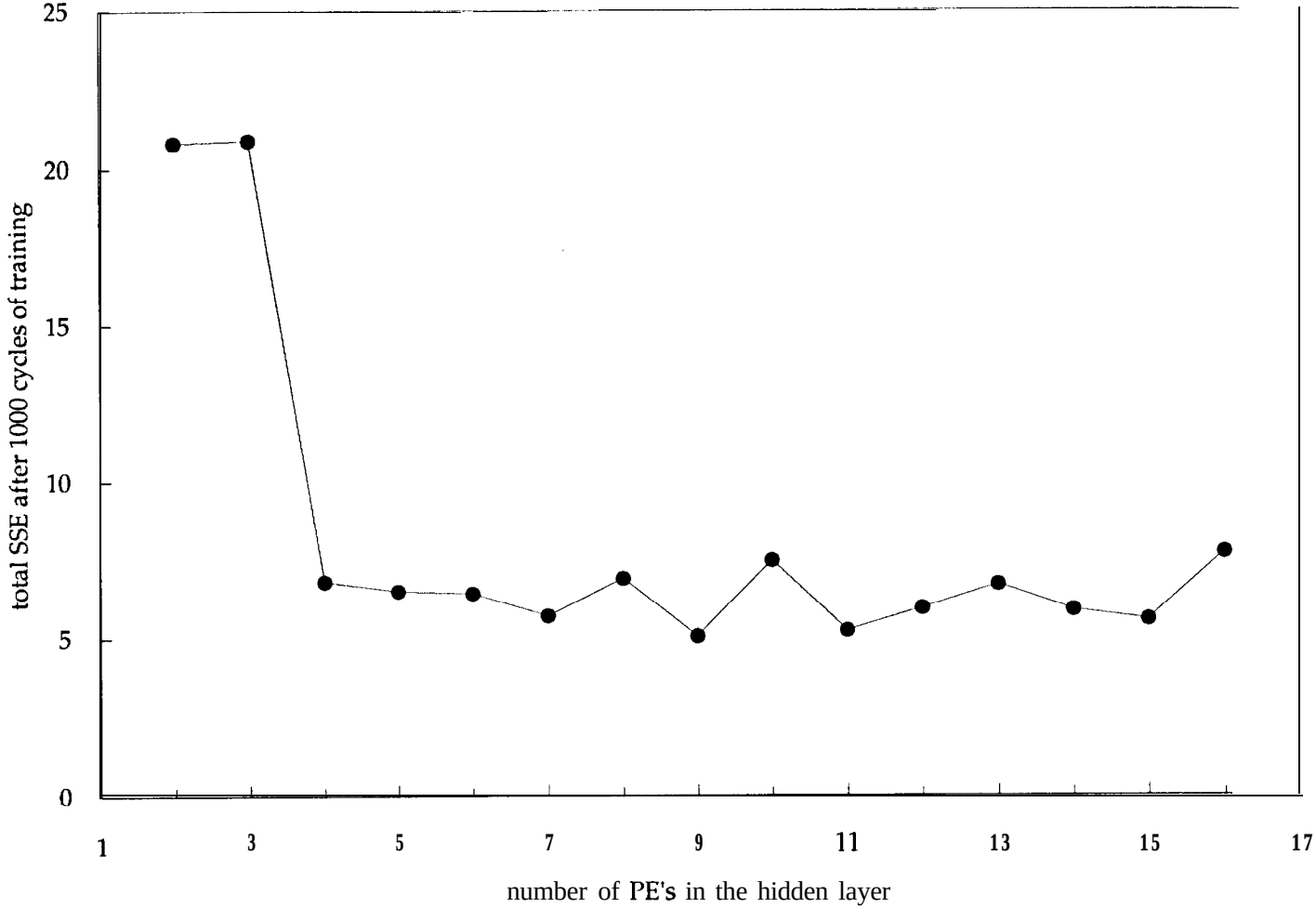

Figure 8.7 SSE for the 2-station 2-interval net #2 with different numbers of hidden PE's

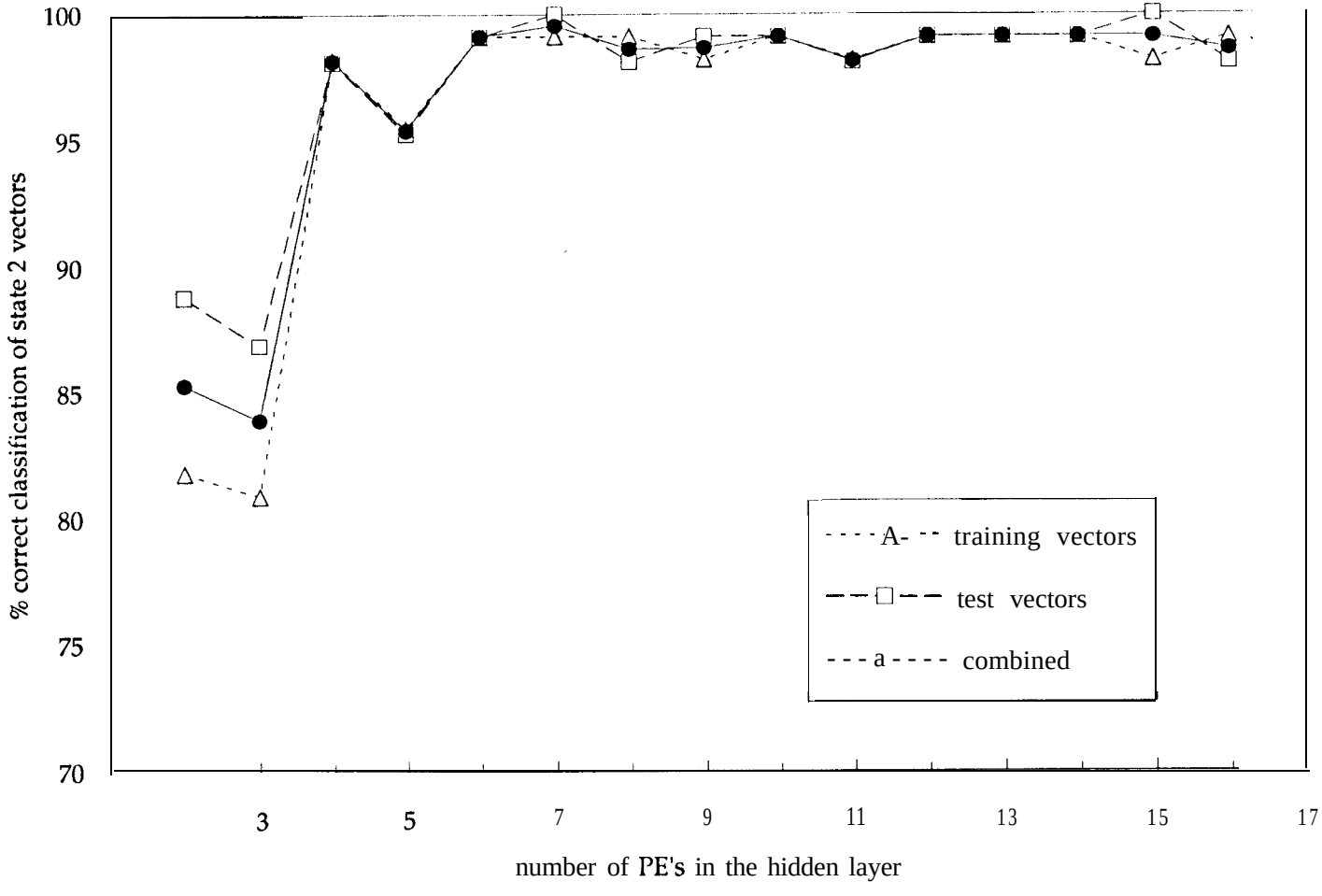

Figure 8.8 Percent correct classification of state 2 vectors for the 2-station 2-interval net #2 with different numbers of hidden PE's

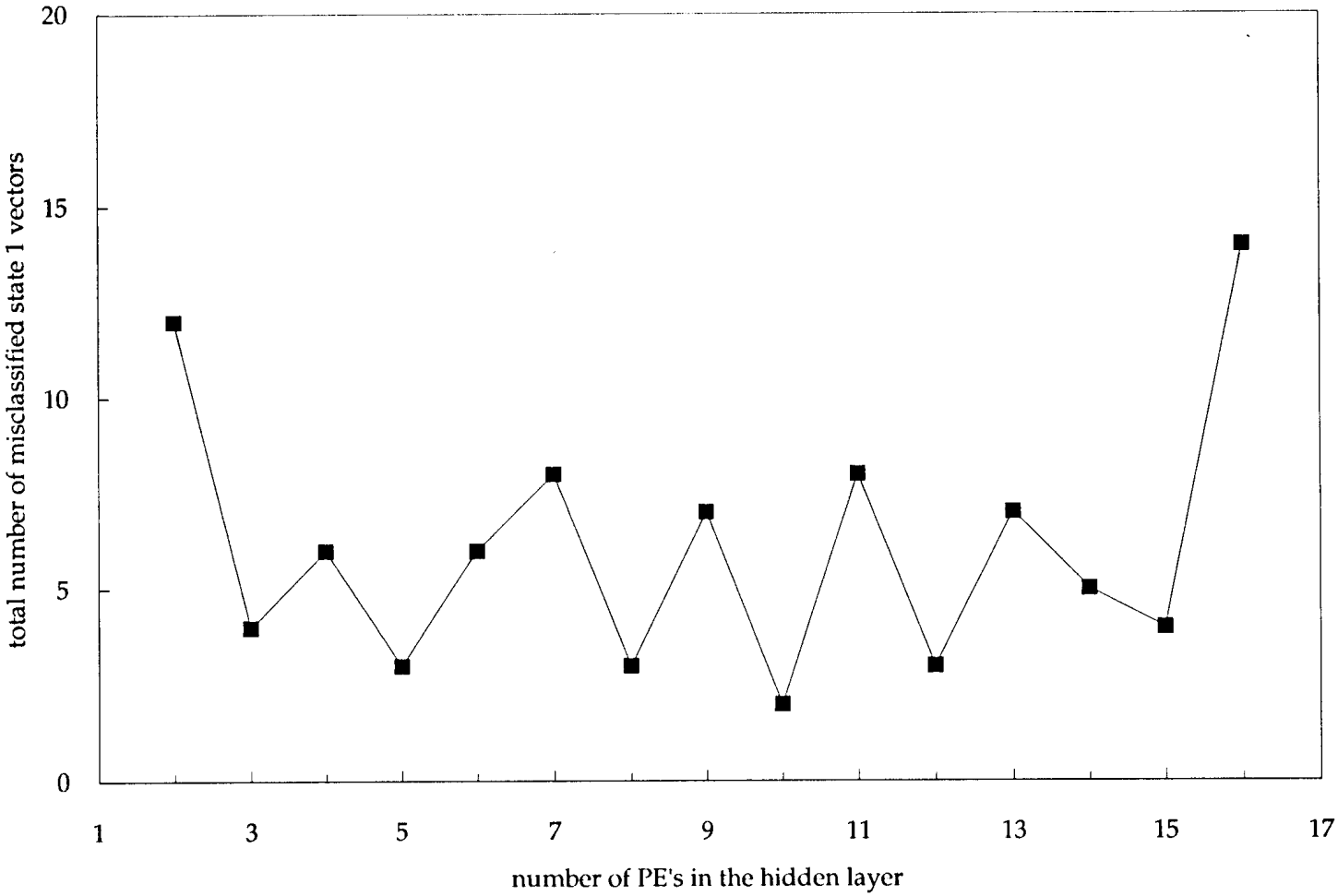

Figure 8.9 Total number of misclassified state 1 vectors for the 2-station 2-interval net #2 with different numbers of hidden PE's

The SSE was reduced, as expected, from 5.700 to 4.204 with more training. The number of misclassified state 1 vectors also decreased from 18 to 16. However, the percentage correct classification of state 2 vectors dropped slightly from 99.5% to 99.1%. This was because one additional state 2 vectors in the training data set was classified as state 1. Since the parameters at 1000 cycles of training had a higher detection rate for state 2 vectors, this network was selected.

8.5 RESULTS OF DOWNSTREAM-STATION THREE-INTERVAL NET

The plot of SSE with different numbers of hidden PE's, each with 1000 cycles of training is shown in Figure 8.10. The SSE has a narrow range in between 20 and 22. The small difference indicates that, despite different numbers of hidden PE's, the nets have in general achieved the same level of performance in classification.

Compared with the models which have 2 station inputs, the downstream-station 3-interval net has much higher SSE at the same number of training cycles. Since the downstream-station 3-interval net accepts input only from 1 station, it lack the capability to distinguish incidents within and upstream of the test section. The network classifyies some training vectors from upstream incidents as state 2, giving rise to a higher SSE.

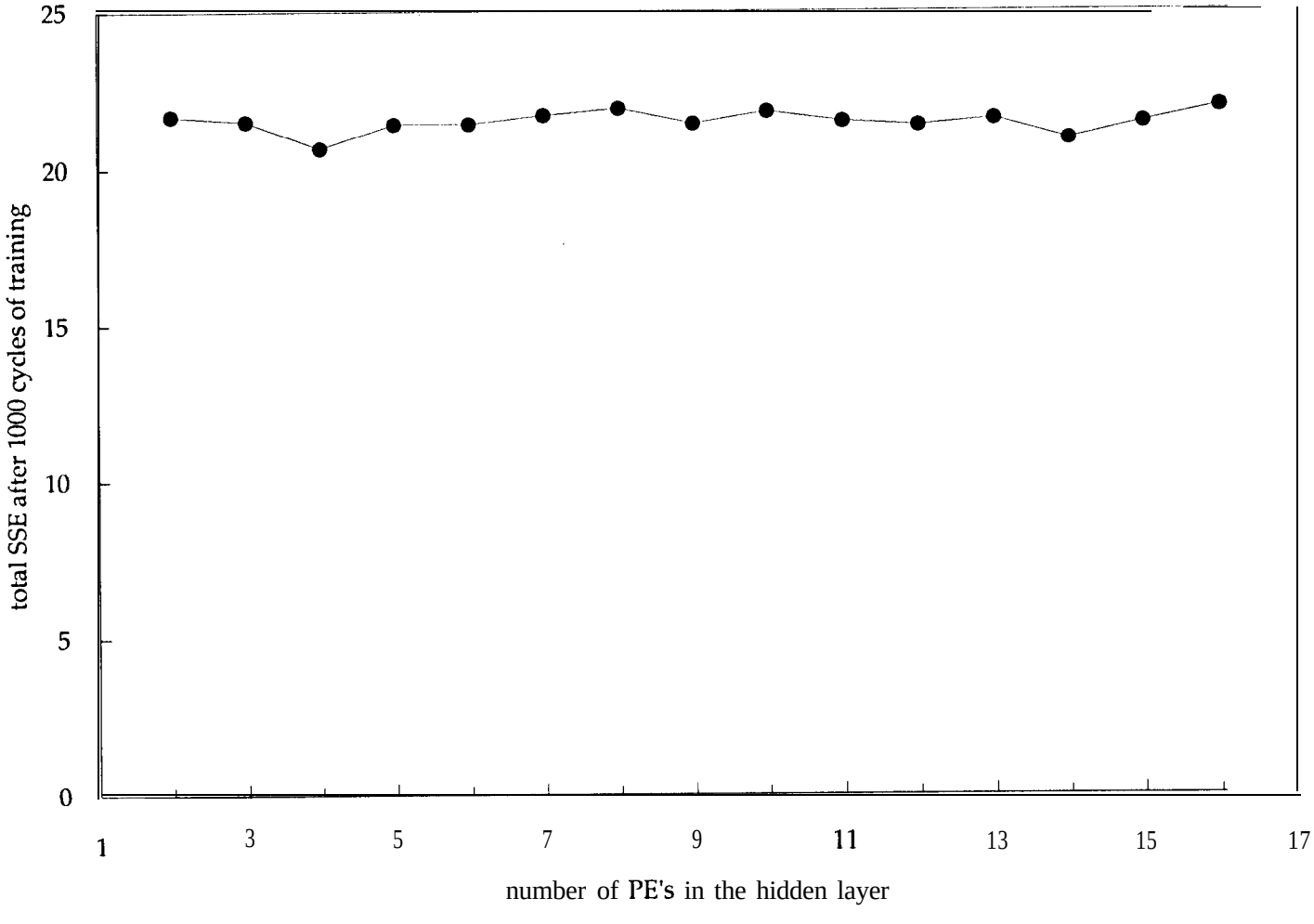

Figure 8.10 SSE for the downstream-station 2-interval net with different numbers of hidden PE's

The percentage correct classification of state 2 vectors in the training, test and the combined training and test data tests is shown in Figure 8.11. Besides the nets with 4 and 14 hidden PE's, most of the networks have about the same level of accuracy in classifying state 2 vectors.

The same trend was also found when applying the networks, with the trained parameters at 1000 cycles, to classify the state 1 vectors. The number of misclassifications of state 1 vectors, plotted in Figure 8.12, was always greater then 40. Those vectors which were misclassified were subsequently inspected. Despite the error in classification, the input features showed that the majority of those vectors had the same pattern of data movement as in Figure 6.1, similar to the beginning of incidents. These vectors were assigned to state 1 because they were from simulation runs where incidents were placed upstream of the test section.

Since the number of misclassified state 1 vectors and SSE were about the same for majority of the trained networks, and there were eight networks with same highest accuracy in classifying state 2 vectors (see Figure 8.11), the decision was made to select the network with the least number of hidden PE's. The network with six hidden PE's was therefore chosen and trained for 2000 cycles. At 1000 cycles of training, this network had a SSE of 21.42, 97.6% overall correct classification of state 2 vectors in both data sets. The training data set has four state 2 vectors classified as state 1, while the corresponding number for the test set was two. There are total of 44 misclassified state 1 vectors for both data sets.

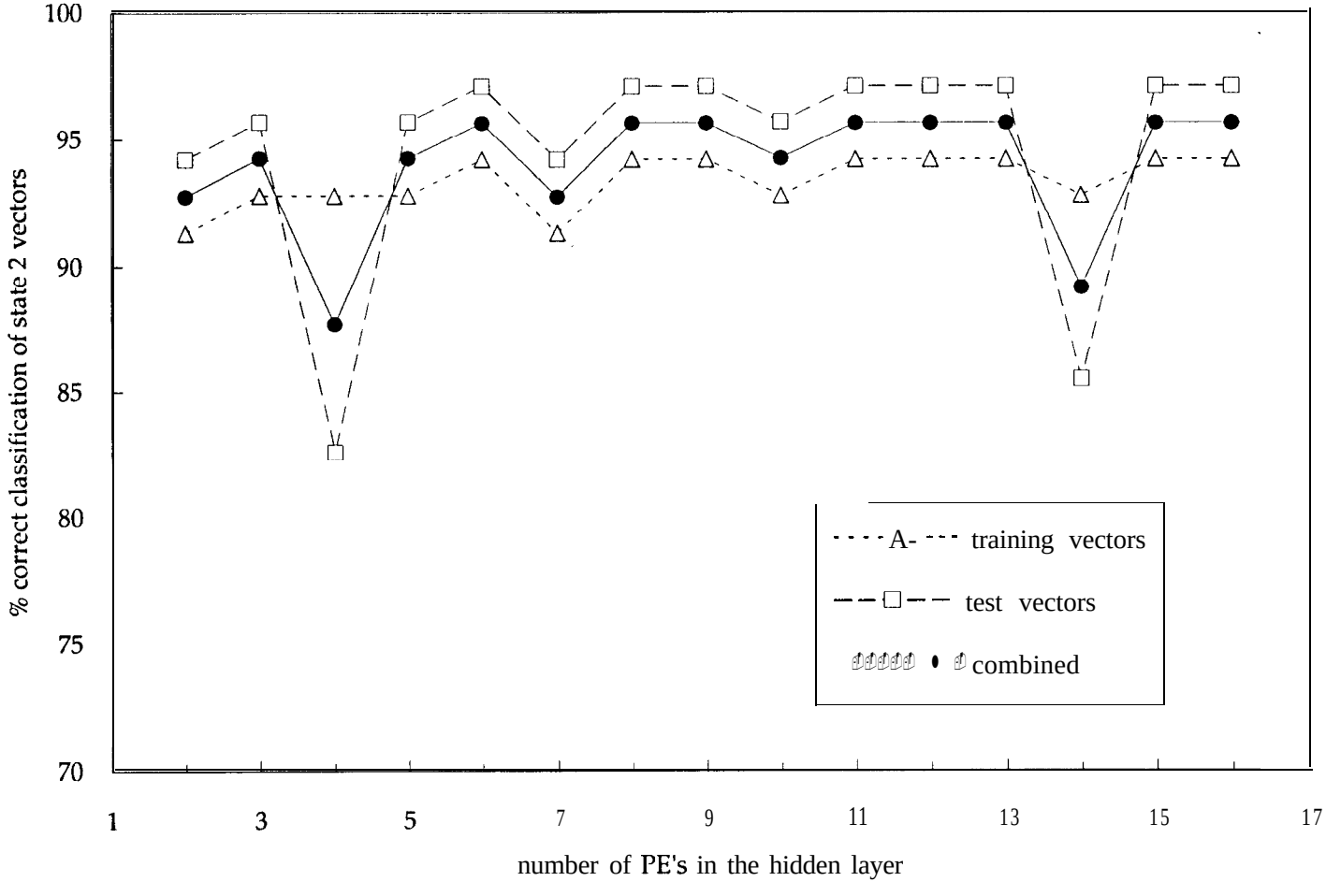

Figure 8.11 Percent correct classification of state 2 vectors for the downstream-station 3-interval net with different numbers of hidden PE's

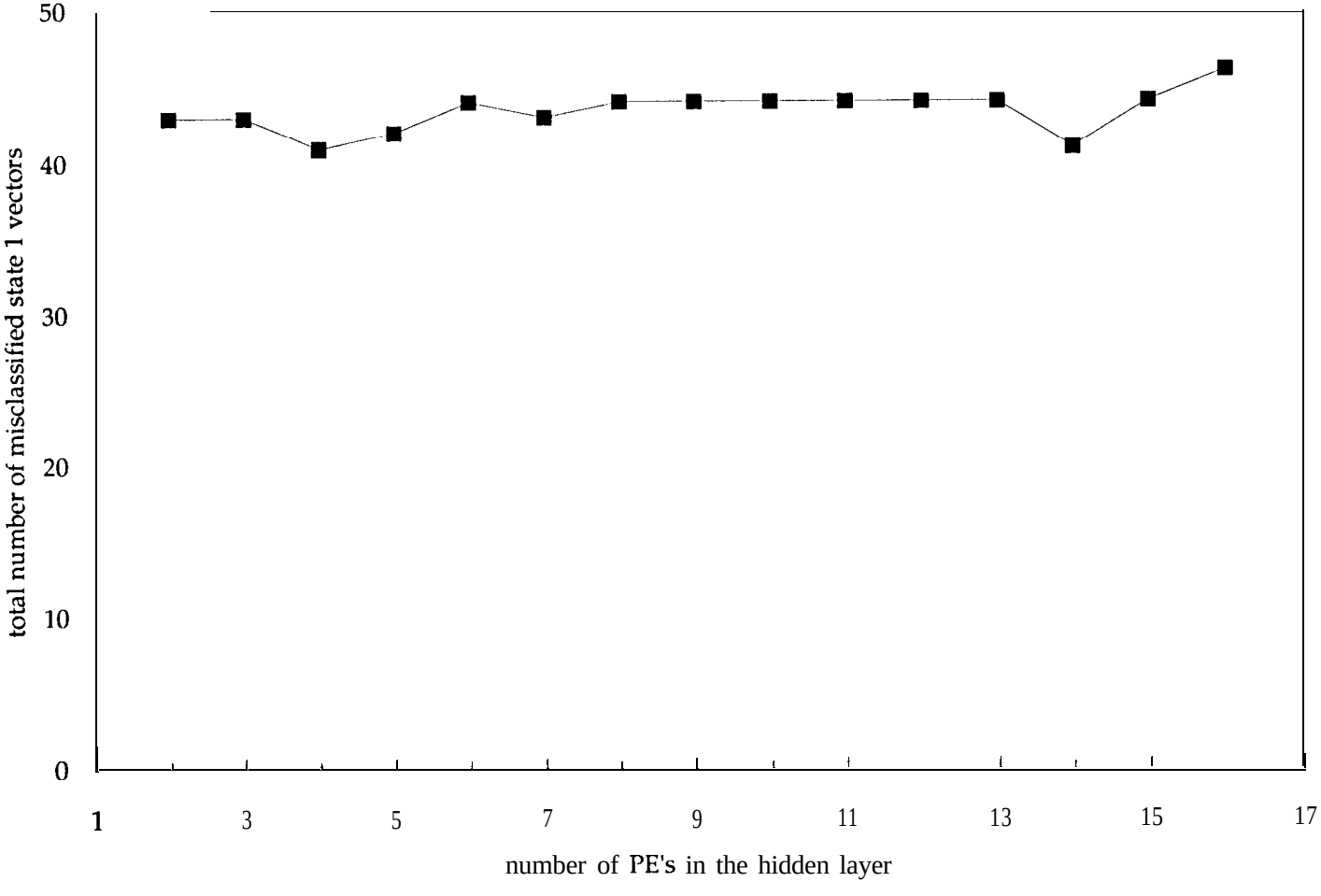

Figure 8.12 Total number of misclassified state 1 vectors for the downstream-station 3-interval net with different numbers of hidden PE's

After 2000 cycles of training, the SSE was reduced slightly to 20.57. The percent correct classification of state 2 vectors remained the same, but one additional state 1 vectors in the test set was classified as state 2. Since the parameters at 1000 cycles of training had fewer misclassification of state 1 vectors, those parameters were used for off-line incident detection evaluation.

8.6 RESULTS OF UPSTREAM-STATION THREE-INTERVAL NET

The upstream-station 3-interval net also faced problems similar to those encountered in the training of the downstream-station 3-interval net. It still had a high SSE after 1000 cycles of training, as shown in Figure 8.13, despite varying the number of hidden PE's. The high SSE was due to the queues formed by incidents downstream of the test section. There were 30 incidents downstream of the test section that are included in the training data set. Some of the incident queues reached the upstream station, causing an increase in occupancy values. Therefore, the associated input vectors were recognized by the nets as state 2, when they should have been state 1.

In this case, the networks were capable of memorizing the mapping between the input and output vectors for state 2 at 90% accuracy in the training set (see Figure 8.14). However, there was at least a 20% reduction in the correct classification rate of state 2 vectors in the test data set. As the inputs to these nets were the upstream station's data, the assignment of

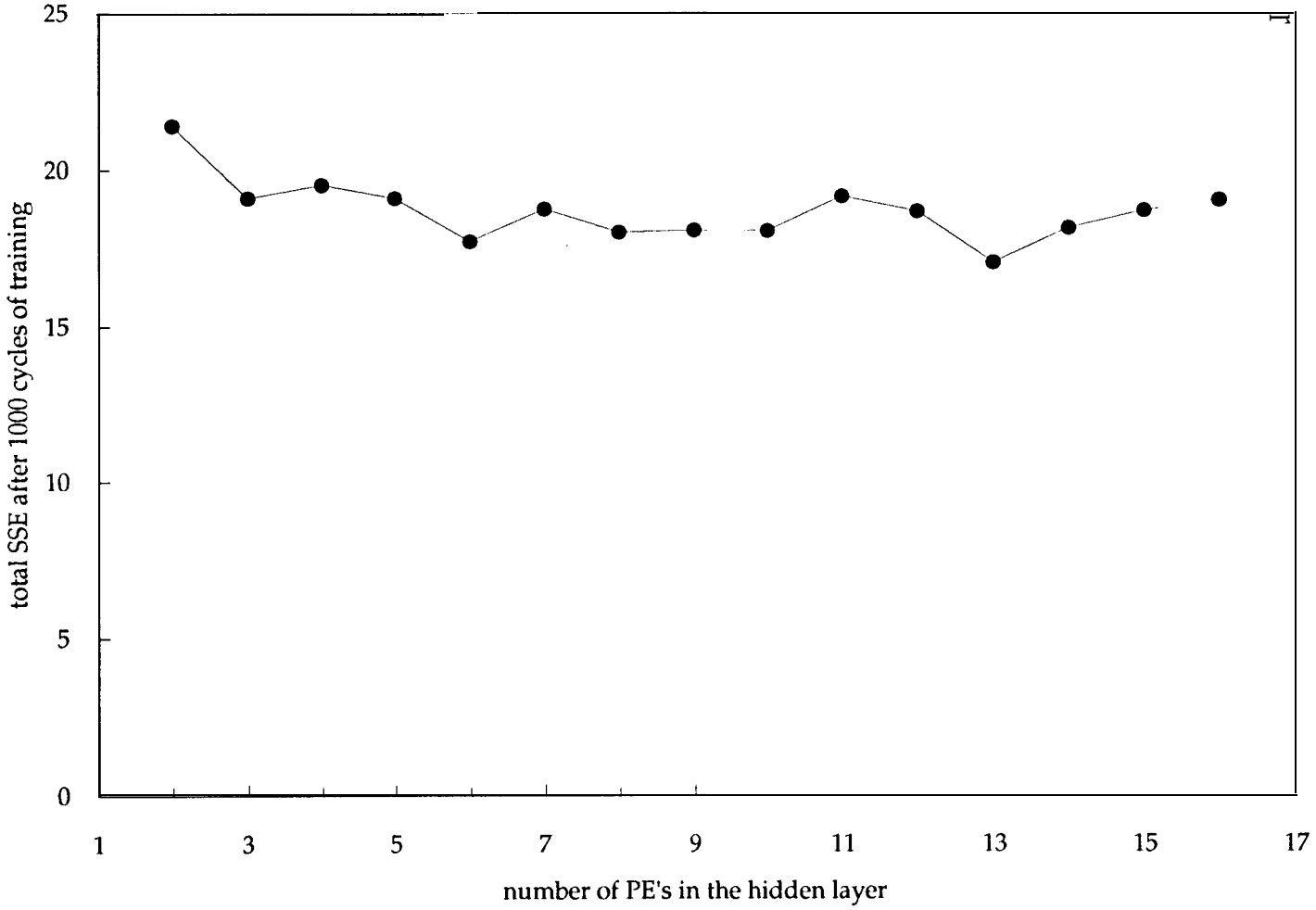

Figure 8.13 SSE for the upstream-station 3-interval net with different numbers of hidden PE's

state 2 to the training and test vectors can only follow an increase in the occupancy at the upstream station. When an incident queue reaches the upstream station, the data point, as illustrated in Figure 6.2, moves to the right hand side of the volume-occupancy plot. In contrast to the the downstream station data, which generally moves towards the origin of the graph, the upstream data point has a much larger region to the right hand side as its destination. Furthermore, as the traffic condition corresponding to this right-hand region is unstable, there is usually a large fluctuation of volume from one interval to another. With the many possibilities for movement of data point in the upstream station, it is not surprising that the network, trained with 90% correct recall of state 2 vectors, was in error when presented with different patterns it had not been trained to recognize. The only remedy to this problem is to increase the size of the training data set (i.e. generate more incidents with different conditions). However, this would correspondingly increase the data formation effort and training time.

The total number of misclassified state 1 vectors in both data sets is shown in Figure 8.15. All values were greater than 28. The test set had more misclassified vectors than the training data set.

In Figure 8.14, the networks with 15 and 16 hidden PE's both have the highest overall correct classification rate of 81.9% for state 2 vectors. Although the number of misclassified state 1 vectors is smaller for 15 hidden PE's, both vectors were selected for further training.

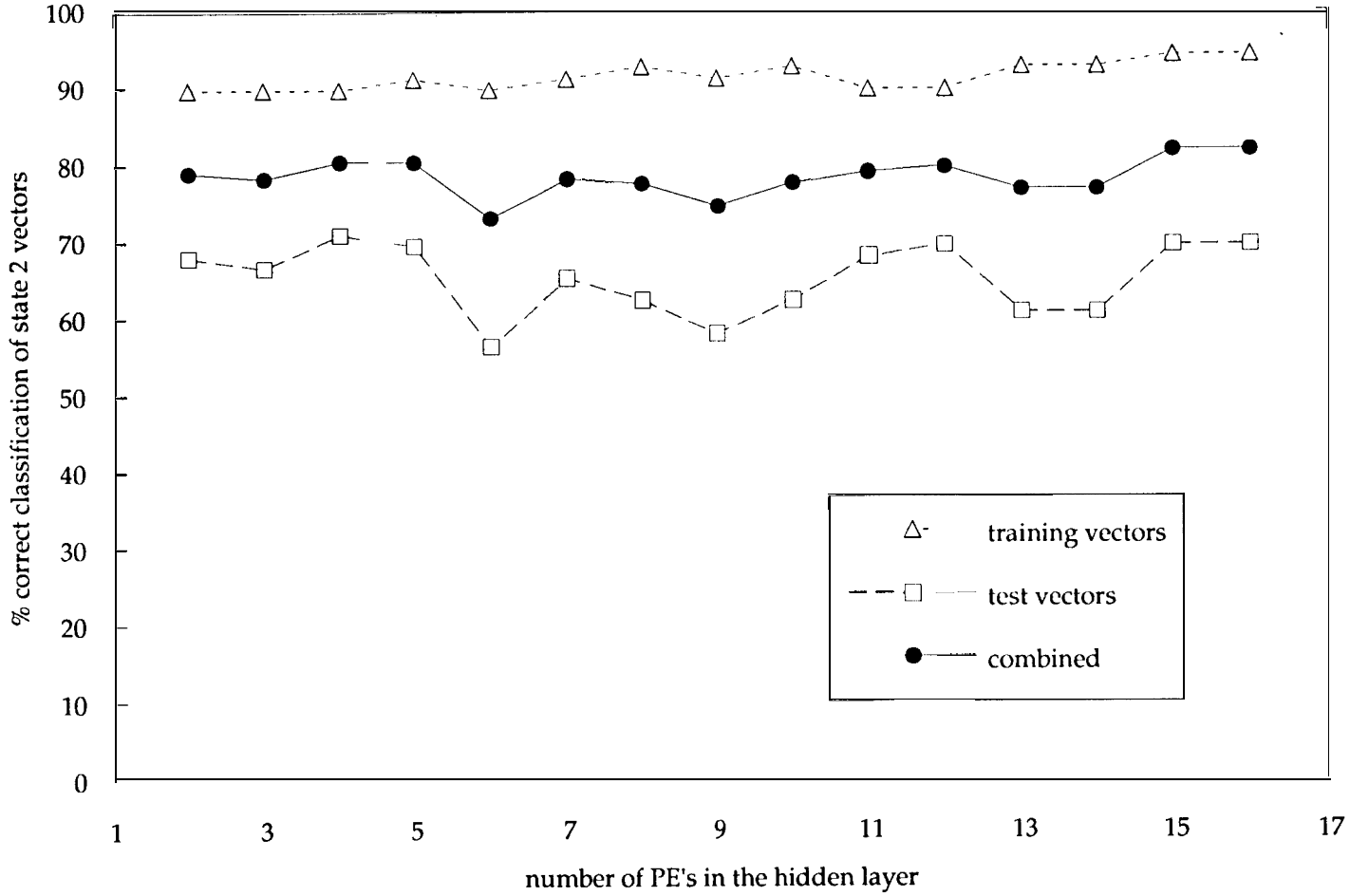

Figure 8.14 Percent correct classification of state 2 vectors for the upstream-station 3-interval net with different numbers of hidden PE's

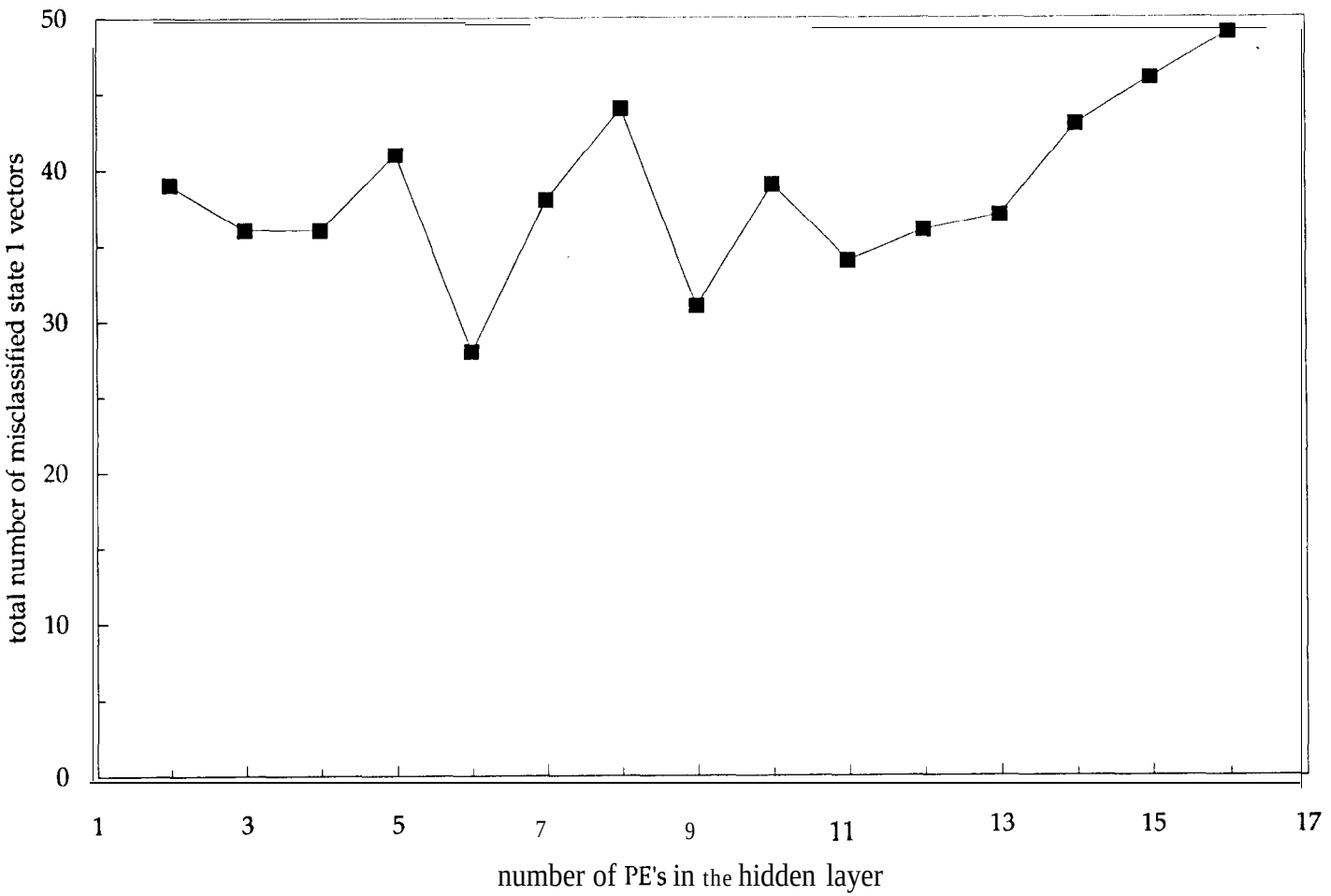

Figure 8.15 Total number of misclassified state 1 vectors for the upstream-station 3-interval net with different numbers of hidden PE's

The results of 1000 and 2000 cycles of training for these two networks are summarized in Table 8.2. When these two nets were subjected to further training, the same level of correct classification of state 2 vectors could not be achieved. The only improvement was in the number of misclassified state 1 vectors for the net with 16 hidden PE's. Based on the best classification rate of state 2 vectors and the smaller number of misclassified state 1 vectors, the net with 15 hidden PE's trained for 1000 cycles was used for off-line incident detection evaluation.

Table 8.2 Summary of training results of upstream-station 3-interval nets

no. of hidden PE's	no. of training cycles	SSE	% correct recall of state 2 vectors on both data sets	# of misclassified state 1 vectors
15	1000	18.689	81.9	46
15	2000	14.536	75.4	47
16	1000	19.005	81.9	49
16	2000	14.341	78.3	37

8.7 CONCLUDING COMMENTS

The five proposed MLF models were trained with the data sets described in Chapter 7, using the backpropagation algorithms. For each model, 15 training sessions, each with a different number of hidden PE's, were performed. After 1000 training cycles, results of the different sessions were compared, and the 'optimum' number of hidden PE's was selected. Training was then continued for the net with the selected number of

hidden PE's, up to 2000 cycles. It was found that, for all the five proposed models, training of up to 2000 cycles did not lead to an improved results. Table 8.3 summarizes the the number of PE's in the hidden layer for the five MLF models. The network parameters (in terms of connection weights and threshold values) obtained after 1000 cycles of training were used to evaluate the off-line incident detection performance.

Table 8.3 Selected number of hidden PE's for the five MLF's after training

MLF models	no. of hidden PE's
2-station 3-interval net	13
2-station 2-interval net #1	14
2-station 2-interval net #2	7
upstream-station 3-interval net	6
downstream-station 3-interval net	15

CHAPTER 9

OFF-LINE INCIDENT DETECTION PERFORMANCE

9.1 INTRODUCTION

The data used in these off-line tests were the simulation cases used to form the test data set. In these tests, the optimal parameters of each of the proposed models were applied to the data sets. The output of the neural net models is a series of states which is used to compute the performance measures.

The off-line test was divided into two parts. In the first part, the results of the five proposed MLF's were compared with each other, and the model with the best incident detection performance was chosen. The second part of the test consists of recalibration of the thresholds of the California algorithm. The California algorithm, with recalibrated thresholds, was then applied to the test data and the results compared with those obtained from the best MLF model.

9.2 MEASURES OF EFFECTIVENESS

The results of the off-line incident detection tests are summarized in the three statistics: detection rate (DR), false alarm rate (FAR), and time-to-detection (TTD). Since the data are extracted from simulation runs, the exact locations and duration of incidents were known, enabling easy computation of the statistics. Each simulation run generated 34 minutes worth of detector data for testing.

An incident located in the test section was labeled "detected" if the the first warning signal (either state 2 from the neural network models or state 7 from the California algorithm) was given within the 5 minute incident period. The detection rate was computed by dividing the total number of detected incidents by the total number of incidents (occur within the test section) in the data set.

The FAR,, as defined in Eq. [2.2], was computed by dividing the false alarms, including repeated warnings during the incidents, by the total number of incident detection tests. The number of incident detection test includes the intervals with correct detection and during incident simulation (which is equivalent to 34 minutes per simulation). Alternatively, repeated warnings during incidents may be taken positively. In this case, FAR,, is found by dividing the total number of alarms outside the five minute incident period by the total number of non-incident intervals (equivalent to 29 minutes per simulation).

The TTD of each correctly detected incident is usually expressed in seconds. However, the California algorithm is applied at 60 second intervals, in contrast to the 30 second interval for the neural net models. Thus, the TTD was also expressed in terms of the number of intervals of application of the model. The first interval occurs in the first application of the model after the occurrence of an incident.

9.3 PERFORMANCE OF NEURAL NET MODELS

Statistics of the test data set are presented in Table 9.1. The 2-station 3-interval net had the highest DR, and lowest FAR, and FAR,. It was the most effective model among the five proposed MLF's. The 2-station 2-interval net #2 had higher FAR's compared to the first net, when the inputs at t-2 were not used. Hence, the input from the downstream station should consist of t, t-l and t-2 intervals for better FAR. The DR of the 2-station 2-interval net #2 was better then the 2-station 2-interval net #1. But it had more false alarms. It can be seen from the difference in FAR, and FAR, that, among the 51 false alarms included in the computation of FAR,, 47 occurred during the incident period. If these repeated warnings are ignored, the rate shown in FAR, is more competitive. However, it is still less attractive than the 2-station 3-interval net. The neural net with only downstream inputs also achieved high DR. However, it had 22 error signals from either definition of false alarms. Among the false alarms, 20 occurred when there were incidents upstream of the test section. If additional logic is implemented to

eliminate such warnings (such as incorporating a rule to check the state at the upstream section), the rates will be reduced to 0.024 and 0.026 for FAR, and FAR, respectively. This level of effectiveness will be at the same level as the above mentioned models. All of the above nets had average TTD of about 68 seconds. The test section is almost one mile in length, which is twice the length of an average section. Shorter TTD can be expected for a shorter freeway section (i.e. shorter distance between detector stations). The upstream-station 3-interval net had the lowest DR, highest FAR and longest TTD. This suggests strongly that upstream stations should not be relied upon solely for incident detection purposes.

Table 9.1 Results of off-line incident detection test for proposed MLF's

network	DR	FAR_1	FAR_2	TTD (second)	TTD (interval)
2-station 3-interval net	100%	0.012%	0.013%	68.5	1.78
2-station 2-interval net #1	97.1%	0.036%	0.039%	68.1	1.77
2-station 2-interval net #2	100%	0.610%	0.052%	69.4	1.81
downstream-station 3-interval net	97.1%	0.263%	0.287%	69.6	1.82
upstream-station 3-interval net	55.1%	0.442%	0.482%	223.0	6.93

9.4 PERFORMANCE OF CALIFORNIA ALGORITHM

The California algorithm #8, or the roll-wave algorithm currently used by Caltrans in Los Angeles freeways, was used as the benchmark algorithm for this study. Several sets of thresholds of this algorithm have been calibrated by Payne et al (1976) using data collected in the early 1970's on three freeways in Los Angeles. When the calibrated thresholds applied to a particular section for incident detection, they can be fine tuned to local traffic flow and patterns of data during incidents (Arcenaeux et *al, 1989).* For fair comparison with the neural network, the thresholds were recalibrated using the 69 incidents within the test section from the training data set.

A program was writen in the C language to perform calibration runs, according to the process documented by Knobel and Heifenbein (1976). Details of the recalibration process are presented in the Appendix of this report. The recalibration yielded a new threshold set of:

T1 = 6.30

T2 = -0.376

T3 = 0.737

T4 = 28.56

T5 = 30.0

The results of applying this algorithm with the recalibrated thresholds are tabulated in Table 9.2. This set of thresholds gives 100%

DR, 0.072% for FAR, and 0.084% for FAR, for the training data set. There were 3 false alarms, due to one inciden upstream and two incidents downstream of the test section. Applying this set of thresholds to the test data set yielded 100% DR, 0.096% for FAR, and 0.112% for FAR,. There were a total of four false alarms in the test data set, of which two came from the 69 incidents in the tests section and 1 each due to upstream and downstream incidents. The TTD for the test data set was 163 seconds, or 2.47 intervals.

Table 9.2 Results of off-line incident detection test for California algorithm with recalibrated thresholds

application interval	data set	DR	FAR,	FAR,	TTD (second)	TTD (interval)
60 seconds	training set	100%	0.072%	0.084%	170.2	2.59
	test set	100%	0.096%	0.112%	163.0	2.47
30 seconds	training set	100%	0.239%	0.208%	128.1	3.77
	test set	98.6%	0.191%	0.182%	130.2	3.84

Because the California algorithm is applied at 60 second intervals instead of 30 seconds as with the neural network models, equal numbers of false alarms will translate into a higher FAR for the California algorithm. The longer interval also means that it has longer TTD. Therefore, the California algorithm was recalibrated for a 30 second application interval, keeping the same structure and input features. The calibrated thresholds for 30 second application were:

T1 = 4.94

T2 = -0.413

T3 = 0.775

T4 = 26.61

T5 = 30.0

This set of thresholds, when applied to the training data set, yielded the following results: 100% for DR, 0.239% for FAR,, 0.208% for FAR, and 128 seconds or 3.77 intervals for TTD. Compared with the 60 second application, the 30 second version had about 35 seconds faster detection time, but has more false alarms. There were 20 and 16 false alarms based on the definition of FAR, and FAR, respectively. Eleven of the false alarms were due to upstream incidents. Similar results were obtained when this set of thresholds was applied to the test data set. Comparisons of the results can be found in Table 9.2.

9.5 COMPARATIVE EVALUATION

The California algorithm and neural net model were calibrated or trained with the training data set. A comparison between these two incident detection models was made using the test data set. The results obtained with the 60 and 30 second versions of the California algorithm were compared with those from the 2-station 3-interval net, the best MLF model. Performance statistics of these models are tabulated in Table 9.3.

Table 9.3 Comparison of incident detection test between California algorithm and the 2-station 3-interval net

network	DR	FAR,	FAR,	TTD (second)	TTD (interval)
2-station 3-interval net	100%	0.012%	0.013%	68.5	1.78
California algorithm at 60 second interval	100%	0.096%	0.112%	163.0	2.47
California algorithm at 30 second interval	98.6%	0.191%	0.182%	130.2	3.84

Both the 60 second California algorithm and the 2-station 3-interval net successfully detected all the incidents in the test data. The California algorithm had much longer TTD. This was because (1) this algorithm is only applied every 60 seconds; and (2) it requires a persistence test of two intervals to declare an incident. The FAR's are apparently eight times higher then the neural net. In fact, there were four false alarms given by

the 60 second California algorithm, and 1 false alarm issued by the neural net model. Since the California algorithm has half the application frequency compared to the neural net model, the FAR is twice as much for the same number of false alarms.

The only false alarms in the test of the Z-station 3-interval net were during an incident with full blockage immediately downstream of station 8 (outside the test section). There was a reduction in volume while the occupancy was increasing at the downstream station. The four false alarms found by the 60 second California algorithm occurred when there were incidents adjacent to the upstream or downstream detector stations. Two of the false alarms were issued after incidents near the downstream end, outside the test section, had been removed. These alarms were due to the remaining portion of incident queues arrive at the upstream station. The other 2 alarms were during incidents that occurred immediately outside the test section.

One way of reducing the TTD, and for better comparison of the FAR, is to apply the California algorithm at 30 second intervals. The 30 second California algorithm detected all but one incident in the test data set. The incident which this algorithm failed to detect occurred under low volume, with a l-lane blockage at the downstream end of the test section. The occupancy values showed that the incident queue only reached the upstream station three minutes after the incident was removed. This algorithm gave higher FAR, with 16 and 14 false alarms for FAR, and FAR, respectively. Eleveen false alarms occurred after incidents upstream of the test section have been cleared. The platoon of vehicles discharge

from incident queues and moving into the test section causes the occupancy at the upstream station to increase, which turned on the incident alarms. The average TTD of the 30 second California algorithm was still about twice as long as that from the Z-station 3-interval net.

9.6 CONCLUDING COMMENTS

Based on the test results, the 2-station 3-interval neural net model appears to be more effiecient than the two California algorithms in detecting incidents, in terms of DR, FAR and TTD. The neural net models could detect at least the same number of incidents as the California algorithm is capable of. It generates one quarter of false alarms compared with the least number given by either version of the California algorithm. The test results show that, on the average, this MLF can detect an incident within 70 seconds from the time of occurrence. The California algorithm took at least twice as long to detect incidents.

CHAPTER 10

CONCLUSIONS AND RECOMMENDATIONS

10.1 CONCLUSIONS

This study has investigated the application of artificial neural networks to the automated detection of lane-blocking incidents on an urban freeway. The relatively simple parallel distributed processing architecture applied, in the form of a multi-layer feed-forward neural network, demonstrated an excellent capability to recognize and classify freeway traffic patterns.

By using different combinations of input features in five different multi-layer feed-forward neural networks and comparing their results, it was found that, to achieve the best results in terms of highest detection rate and lowest false alarm rate, it was necessary to use the station average volume and occupancy values at the upstream station for the past two intervals, in addition to the upstream and downstream data for the current interval.

Based on data obtained from a microscopic freeway traffic simulation model and comparisons with the California incident detection algorithm with calibrated thresholds, the results obtained in this research suggest that neural network models have potential to achieve significant improvements in incident detection performance.

10.2 **RECOMMENDATIONS**

Based on experience gained in this study and the results obtained, recommendations further research are as follows:

(1) Although real data would be preferred, it appears that the neural network models will have to continue for some time to be trained on loop detector data generated from INTRAS. The simulation model should therefore be calibrated to produce output closely matches real detector output obtained from the testbed.

(2) Efforts should continue to collect as many real incident data sets as possible. Both neural network models and conventional incident detection algorithms should be tested with such data.

(3) The incident detection capability of other types of neural network models, such as the self-organizing feature map, for example, should be investigated and compared with the performance of the multi-layer feed-forward neural network.

(4) Neural network models should be trained in the future with data from multiple freeway sections in the testbed so that a generic model may be developed and used to detect incidents in any freeway section.

(5) The encouraging performance of artificial neural network models for freeway incident detections suggests that the application of such

models to detection of operational problems on arterial streets should also be investigated.

REFERENCES

Ahmed, S. A., (1983), "Stochastic Process in Freeway Traffic", Traffic Engineering and Control", Vol. 24, No. 6, pp. 306-314.

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

Akamatsu, T., Tsuchiya, Y. and Shimazaki, T., (1990), "Parallel Distributed Processing on Neural Network for Some Transportation Equilibrium Assignment Problems", Proceedings of the 10th International Symposium of Transportation and Traffic Theory, Edited by Koshi, M., Elsevier Science, pp. 307-323.

Arceneaux, J., Smith, J., Dunnett, A. and Payne, H., (1989), Calibration of Incident Detection Algorithms for Operational Use. in Traffic Control Methods, Proceedings of the Engineering Foundation Conference, edited by Yagar, S. and Rowe, E., United Engineering Trustees Inc., pp. 17-32.

Aultman-Hall, L., Hall, F. L., Shi, Y., and Lyall, B., A Catastrophe Theory Approach to Freeway Incident Detection, Proceedings of the Second International Conference of Applications of Advanced Technologies in Transportation Engineering, Edited by

Stephanedas, Y. J. and Sinha, K. C., American Society of Civil Engineers, pp. 373-377.

Blosseville, J. M., Motyka, V., Djemame, N. and Lenoir, F., (1991), "Automatic Incident Detection Using Image Processing Techniques: A Specific System Used in INVAID", Proceedings of the Second International Conference of Applications of Advanced Technologies in Transportation Engineering, Edited by Stephanedas, Y. J. and Sinha, K. C., American Society of Civil Engineers, pp. 383-387.

Collins, J. F., Hopkins, C. M. and Martin, J. A., (1979), Automatic Incident Detection - TRRL ALgorithms HIOCC and PATREG, TRRL Supplementary Report 526, Transportation and Road Research Laboratory.

Cremer, M. and Schutt, H., (1990), "A Comprehensive Concept for Simultaneous State Observation, Parameter Estimation and Incident Detection", Proceedings of the 10th International Symposium of Transportation and Traffic Theory, Edited by Koshi, M., Elsevier Science, pp. 95-111.

Fambro, D. B. and Ritch, G. I'., (1979), "Automatic Detection of Freeway Incidents During Low Volume Conditions", Report FHWA/TX-79/23+210-l, Texas Transportation Institute, Texas A&M University.

Gall, A. I., and Hall, F. L., (1989), "Distinguishing between Incident Congestion and Recurrent Congestion: A Proposed Logic", Transportation Research Record 1232, pp. 1-8.

Gan, M. and Liu, X., (1991) "Neural Networks in Structural Preliminary Design", Artificial Intelligence and Civil and Structural Engineering, edited by B. H. V. Topping, Civil-Comp Press, pp. 285-293.

Goldblatt, R. B., (1980), Development and Testing of INTRAS, a Microscopic Freeway Simulation Model, Vol. 3 Validation and Application, Report No. FHWA/RD-80/108, Federal Highway Administration.

Hajela, P., Fu, B., Berke, L., (1991), "ART Networks in Automated Conceptual Design of Structural Systems", Artificial Intelligence and Civil and Structural Engineering, edited by B. H. V. Topping, Civil-Comp Press, pp. 263-271.

Han, L. D. and May, A. D., (1988), Artificial Intelligence Approaches for Signalized Network Control, Working Paper UCB-ITS-WP-88-4, Institute of Transp. Studies, University of California Berkeley.

Han, L. D. and May, A. D., (1989), Automatic Detection of Traffic Operational Problems on Urban Arterials, Research Report UCB-ITS-RR-89-15, Institute of Transportation Studies, University of California, Berkeley.

Judycki, D. C. and Robinson, J. R., (1992), "Managing Traffic during Nonrecurring Congestion", ITE Journal, Vol. 62, No. 3, pp. 21-26.

Kaseko, M. S. and Ritchie, S. G., (1991), "Pavement Image Processing Using Neural Networks", Proceedings of the Second International Conference of Applications of Advanced Technologies in Transportation Engineering, Edited by Stephanedas, Y. J. and Sinha, K. C., American Society of Civil Engineers, pp. 378-342.

Knobel, H. C. and Helfenbein, E. D., (1976), Dvelopment and Testing of Incident Detection Algorithms, Vol. 4: Program Documentation for Calibration, Evaluation and Implementation, Report No. FHWA-RD-76-22, Federal Hignway Administration.

Kohonen, T., (1982), "Self-Organized Formation of Topologically Correct Feature Maps", Biological Cybernetics, Vol. 43, pp. 59-69.

Kohonen, T., (1990), "Some Practical Aspects of the Self-Organizing Maps", Proceedings of the International Joint Conference on Naural Networks, Washington D. C., Vol. II, pp. 253-256.

Kosko, B., (1987), "Adaptive Bidirectional Associative Memories", Applied Optics, Vol. 26, pp. 4947-4960

Levin, M. and Krause, G. M., (1978), "Incident Detection: A Bayesian Approach", Transportation Research Record 682, pp. 52-58.

Levin, M. and Krause, G. M., (1979), "Incident Detection Algorithms. Off-Line Evaluation", Transportation Research Record 722, pp. 49-64.

Lindley, J. A., (1987), "Urban Freeway Congestion: Quantification of the Problem and Effectiveness of Potential Solutions", ITE Journal, Vol. 57, No. 1, pp. 27-32.

Lo, Z-P. and Bavarian, B., (1991), A Neural Piecewise Linear Classifier for Pattern Classification", Proceeding of the International Joint Conference in Neural Network, Vol. 1, pp. 264-268.

Moselhi, O., Hegazy, T. and Fazio, I'., (1991) "Markup Estimation Using AI Methodology", Artificial Intelligence and Civil and Structural Engineering, edited by B. H. V. Topping, Civil-Comp Press, pp. 257-266.

Nakatsuji, T. and Terutoshi, T., (1990), "Application of Neural Network Model to Traffic Engineering Problem: Self-Organising Approach to Traffic Management System", Proceedings of the 10th International Symposium of Transportation and Traffic Theory, Edited by Koshi, M., Elsevier Science, pp. 291-306.

Payne, H. J. and Helfenbein, E. D., (1976), Freeway Surveillance Data, Report No. FHWA-RD-76-74, Federal Highway Administration.

Payne, H. J., Helfenbein, E. D. and Knobel, H. C., (1976), Development and Testing of Incident Detection Algorithms, Vol. 2: Research Methodology and Results, Report No. FHWA-RD-76-20, Federal Highway Administration.

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

Persaud, B. N., Hall, F. L. and Hall, L. M., (1990), "Congestion Identification Aspects of the McMaster Incident Detection Algorithm", Transportation Research Record 1287, pp 167-175.

Ramirez M. R. and Arghya, D., (1991) "A Faster Learning Back-Propagation Naural Networks in NDE Applications", Artificial Intelligence and Civil and Structural Engineering, edited by B. H. V. Topping, Civil-Comp Press, pp. 275-283.

Rosenblatt, F., (1962), Princlples of Neurodynamics, Spartan Books.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J., (1986), "Learning Internal Representations by Error Propagation", in Parallel and Distributed Processing, edited by Rumelhart, D. E., McClelland, J. L. and the PDP Research Group, Vol. 1, MIT Press, pp. 318-362.

Rumelhart, D. E. and Zipser, D., (1986), "Feature Discovery by Competitive Learning", in Parallel Distributed Processing, edited by Rumelhart,

D, McClelland and the PDP REsearch Group, Vol. 1, MIT Press, pp. 151-193.

Sejnowski, T. J. and Rosenberg, C. R., (1986), "NETtalk: A Parallel Network that Learns to Read Aloud", Electrical Engineering and Computer Science Technical Report JHU/EECS-86/01, John Hopkins University.

Sellam, S., Boulmakoul, A and Pierrelee, J. C., (1991), "A Distributed Real TIme Knowledge-Based System Using Video Image Processing for Junctions Automatic Incident Detection", Proceedings of the Second International Conference of Applications of Advanced Technologies in Transportation Engineering, Edited by Stephanedas, Y. J. and Sinha, K. C., American Society of Civil Engineers, 388-392.

Simpson, P. K., (1990), Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementation, Pergamon Press.

Skabardonis, A., Cassidy, M., May, A.D. and Cohen, S., (1989), "Application of Simulation to Evaluate the Operation of Major Freeway Weaving Sections", Transportation Research Record 1225, pp. 91-98.

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

Tsai, J. and Case, E. R., (1978), "Development of Freeway Incident-Detection Algorithms by Using Pattern Recognition Techniques", Transportation Research Record 722, pp. 113-116.

Wasserman, P. D., (1989), Neural Computing, Van Nostrand Reinhold Int. Co. Ltd..

Wicks, D. A. and Andrews, B. J., (1980), Development and Testing of INTRAS, a Microscopic Freeway Simulation Model, Vol. 2, User's Manual, Report No. FHWA/RD-80/107, Federal Highway Administration.

Willsky, A. S., Chow, E. Y., Gershwin, S. B., Greene, C. S., Houpt, P. K. and Kurkjian, A. L., (1980), "Dynamic Model-Based Techniques for the Detection of Incidents on Freeways", IEEE Transactions on Automatic Control, Vol. 25, No. 3, pp. 347-360.

APPENDIX A

CALIFORNIA ALGORITHM RECALIBRATION PROCEDURE

A.1 THE CALIBRATION PROGRAM

The original calibration procedure for the California algorithm reported by Knobel and Helfenbein (1976) was implemented in a program in the C language. The procedure searches for an optimum threshold vector that minimizes the FAR while maintaining a minimum DR. The structure of the calibration program is described in the following paragraphs. The program listing is attached in Appendix B of this report.

The flow chart of the main calibration module is shown in Figure A.l. When the calibration program is executed, the initial feasible solution vector for the thresholds is read, together with other calibration parameters such as the minimum detection rate (min_DR), maximum number of iterations (max_iteration), step size (step-size), and the maximum allowable FAR (max_FAR). To check the feasibility of the initial solution supplied by the user, the California algorithm is applied to the calibration data set using the initial solution vector for the thresholds. The DR obtained is compared with the min_DR. If the DR is less than the minimum requirement, the program is terminated and the user has to restart the calibration process by supplying another initial solution vector. Otherwise, the optimization module is invoked and this module seeks a

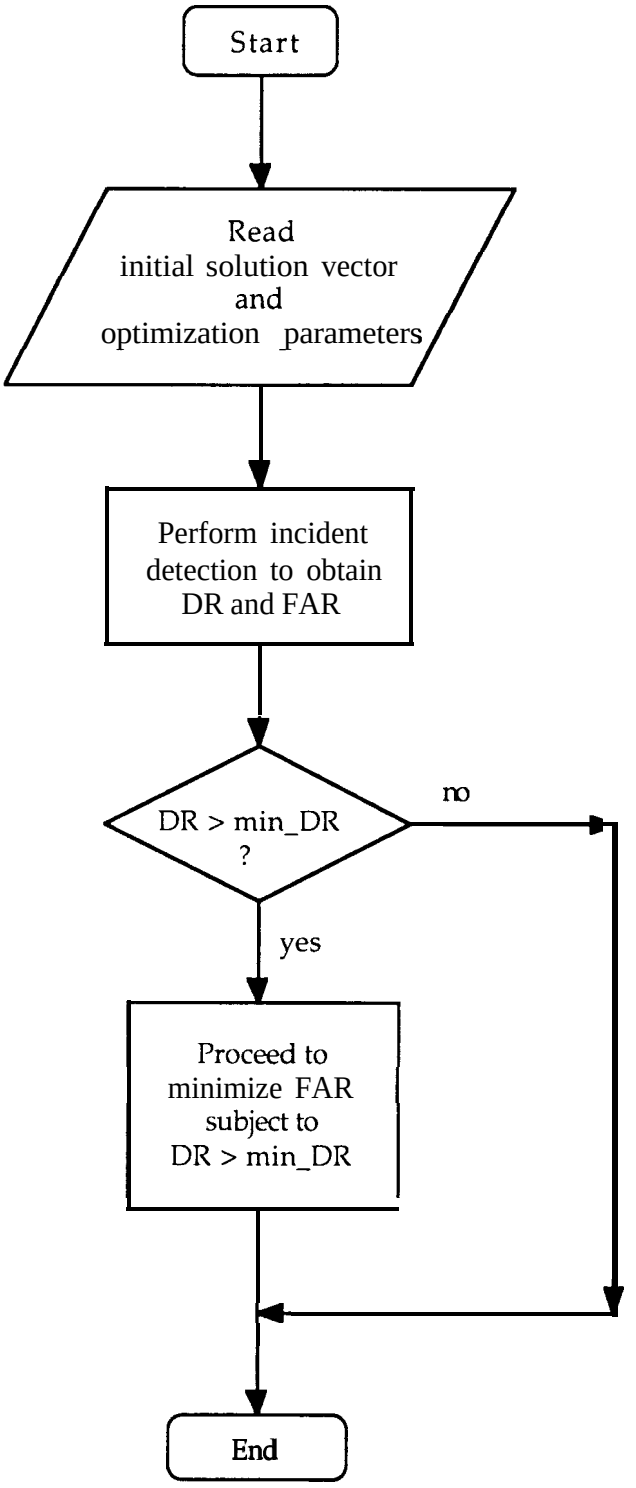

Figure A.1 Flow chart of main calibration module

threshold vector that minimizes the FAR and at the same time keeps the DR greater than the min_DR. The optimization process is shown in Figure A.2.

At the beginning of the FAR optimization module, the FAR of the initial solution is assigned to the 'current' FAR (old-FAR) from which improvement is to be made. The iteration number (i), which counts the number of new solutions that have improved the FAR, is set to 0. The module continues to search for an optimum vector when i is less than max_iteration, and the FAR computed from the current solution is greater than the max_FAR, and when step-size is less than the minimum allowable value (min_step_size).

The search for an optimum threshold vector mainly consists of the generation of new candidate solution vectors and the testing of them against the old_FAR and min_DR criteria. A new candidate solution vector X is obtained by adding a random vector AX to the current solution. Initially, each component of AX is generated from a standard normal distribution. The random numbers are then normalized to form a unit vector. Since different components of the vector have different numerical scales due to the different features in the California algorithm that they represent, they are multiplied by their respective scale factors. Finally, the entire vector is multiplied by a common factor, the step-size, before being added to the current solution to form X.

The new vector X is used for the thresholds in incident detection runs for all the incidents in the calibration data set. If there is an

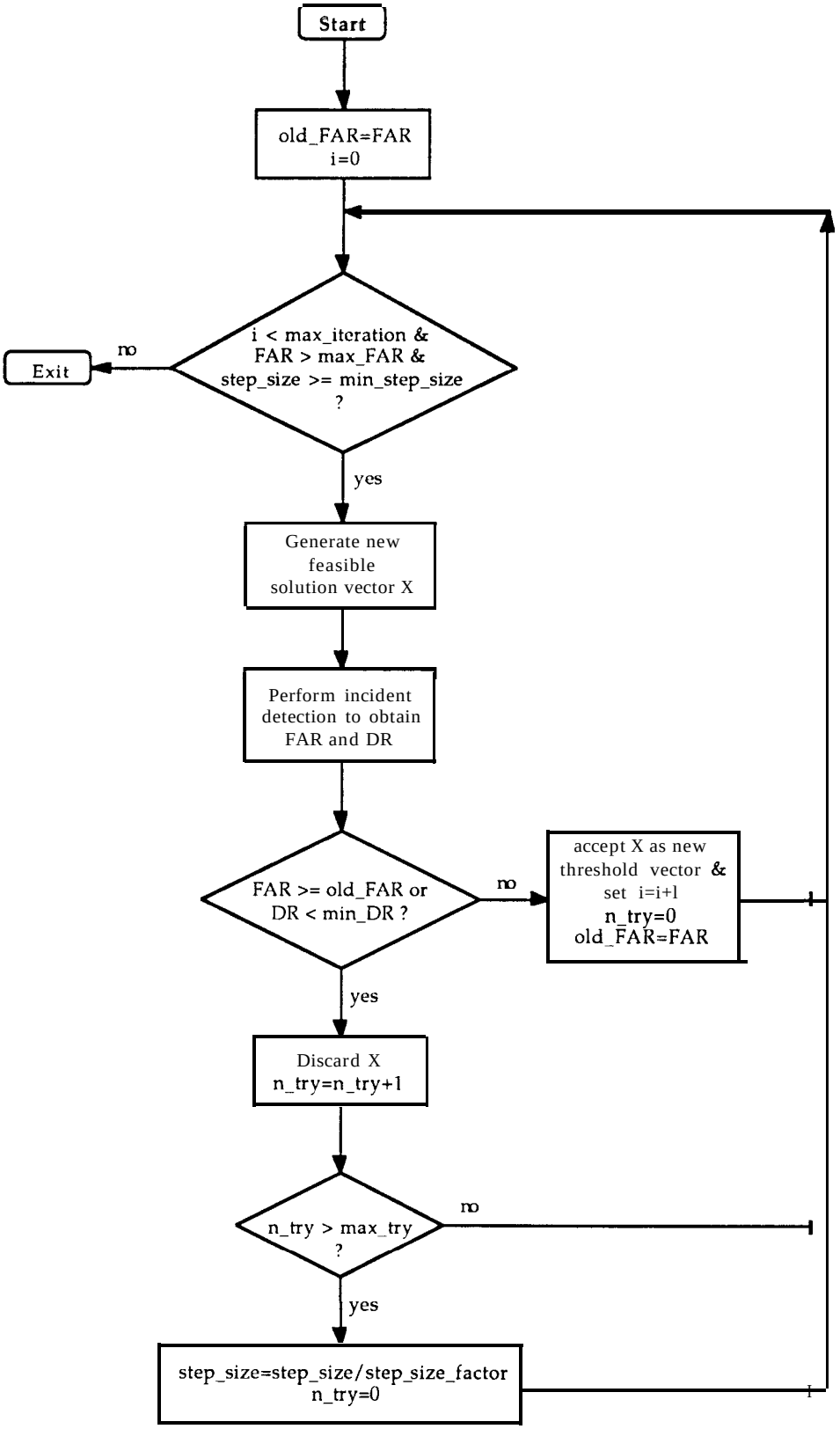

Figure A.2 Flow chart of FAR minimization module

improvement in the FAR from the old_FAR while the DR is greater than min_DR, X is accepted as the new solution. The old_FAR is updated with the new FAR value, the number of iteration, i, is increased by 1, and the number of the trials for X as a new solution is set to 0.

If the X fails to give a lower FAR or the DR is too small, X is discarded with the n-try counter increased by 1. A new candidate vector is next generated and tested. The generation and testing of X is repeated until the maximum number of trials (max_try) is exhausted without an improvement in the performance. At this stage the step-size, which controls the magnitude of the AX vector, is reduced by dividing it by a factor (step-size-factor) and the search process continues. The optimization module can only be terminated when the maximum number of accepted improvements has been reached (i=max_iteration), or when the FAR has fallen below the maximum allowable level (FAR<=max_FAR), or when the step size for the search becomes too small.

A.2 CALIBRATION OF THRESHOLDS FOR 60 SECOND APPLICATION INTERVALS

The original California algorithm was developed for 1 minute average detector data, and is applied at 1 minute intervals. Detector output from INTRAS, in the form of 30 second average occupancies, were aggregated into 1 minute average values for calibration and testing of the California algorithm.

Payne et *al* (1976) has recommended five sets of thresholds for the California algorithm, calibrated with Los Angeles freeway data collected in the period between 1974 and 1976. The first threshold set, which was intended to yield the highest detection rate and shortest time to detection, but which also has the highest false alarm rates, was used as the initial solution in this calibration. This threshold set was selected because the shortest time to detection would give a competitive performance with the MLF's. The values in this threshold vector are (10.20, -0.443, 0.312, 28.8, 30.0). Since the last threshold value is always fixed as 30 (as of the 4 other threshold sets calibrated by Payne), only the first four components are varied in the calibration. Payne et *al* (1976) used only 'incident data' set for their calibration effort, and that set of data did not contain incident free period.s From the training data set which was used to train the MLF's, occupancy values from the 69 simulation runs which consisted of incidents within the test section were extracted and converted into the recalibration data set. Since the California algorithm has the ability to detect the continuation of an incident after its first detection, the repeated

warning of state 7 was not expected and therefore FAR, was computed. The input parameters to the recalibration program were as follows:

min_DR = 80%

max_FAR = 1%

initial step-size = 2.0

min_step_size = 0.01

step-size-factor = 2.0

max_try = 5

max_iteration = 10

scale vector = {1.0, 0.1, 0.1, 1.0, 0.0}.

The initial solution resulted in a DR of 88.4% and FAR, of 1.173%. After generating and testing of 37 solution vectors, the programs accepted 6 improvements. At the 5th trial (n_try=5) of the 7th iterations (i=7), the threshold vector converged to (6.30, -0.376, 0.737, 28.56, 30.0) which gave 100% in DR and 0% for FAR,. This threshold set enabled the California algorithm to detect all the 69 incidents in the recalibration data set without any false alarms.

A.3 CALIBRATION OF THRESHOLDS FOR 30 SECOND APPLICATION INTERVALS

Because the neural network models are designed to be applied at 30 second intervals, the California algorithm was also calibrated for 30 second operation. All the input features in the California algorithm were retained (i.e. the inputs were still based on the average occupancy values for the past 1 minute). The roll-wave suppression logic within the algorithm was still applied for 5 minutes. Only the frequency of the application of the algorithm to incident detection was doubled.

As the recalibrated threshold set of (6.30, -0.376, 0.737, 28.56, 30.0) gave the perfect detection and false alarm rates for 60 second intervals, it was decided to use this vector as the initial solution, and only fine-tuning was expected. The same set of calibration parameters, except the step-size-factor, were used as the input. The step-size-factor was set to 1.5 for the expected minor adjustment.

The initial solution gave DR of 100%, but had FAR, of 9.1%. After testing 20 candidate solution vectors and accepting three improvements, the FAR, was reduced to 0.375% while keeping a 100% DR. The program continued to generate another 73 candidate vectors but all of them gave the same performance statistics. This calibration run was terminated when the step-size was equal to 0.078, less then the minimum value specified. Since there were many vectors that produced the same results,

the last threshold set was taken as the optimum threshold. This threshold vector was {4.94, -0.413, 0.775, 26.61, 30.0}.

To avoid being trapped in a local minimum point in the search for an optimum vector, several trials were made with different initial solutions. However, none of them achieved FAR, of less then 0.375%.

APPENDIX B

LISTING OF THE CALIFORNIA ALGORITHM CALIBRATION PROGRAM

```
/* this program calibrates the CA #8 Roll Wave Surpression Algorithm */
/* writen by Kelvin Cheu on e-1-91 */

#include <stdio.h>
#include <math.h>
#define phi 3.1415927

float rand();
float normal();
void normalize();
void add();
void read_data();
void read_ini_soln();
void inc1();
void get_dx();
void scale_dx();
void copy();

main()
        i
        FILE *fout;
        long seed;
        int i, j, max_i, n_try, max_try, n_detect, n_alarm;
        float occ[150][68][4], dr, min_dr, far, min_far, old_far, th[5],
                scale[5], x[5], dx[5];
        float step_size, min_step, step_div;


        /* read data */
        read_data(occ);
        read_ini_soln(th, scale, &min_dr, &min_far, &step_size, &min_step,
                        &step_div, &max_try, &max_i, &seed);

        /* test initial solution for detection rate */
        n_detect=0;
        n_alarm=0;
        for (i=0;i<123;i++)

                inc1(occ, i, t&n& _detect, &n_alarm);
                }
        dr=((float)n_detect)/69.0;
        far=((float)n_alarm)/8019.0*100.0;
        printf("ini soln x= %5.2f  %6.3f  %5.3f  %5.2f  %4.1f\n",
                        th[0], th[1], th[2], th[3], th[4]);
        printf ("n_detect=%2d  DR=%5.3f  n_alarm=%3d  FAR=%5.3f%\n",
                        n_detect, dr, n_alarm, far);
```

```
if (dr>=min_dr)
        {
        printf("DR passes the minimum value\n");
        printf("proceed to minimize FAR\n");
        copy (th,x, 5) ;
        old_far=far;
        n_try=0;
        i=0;

        while(i<=max_i && old_far>min_far && step_size>=min_step)
                i
                /* get new temporary solution point x */
                seed=seed+3L;
                get_dx (dx, seed, 4);
                normalize(dx,4);
                scale_dx(dx,scale,step_size,4);
                add(x,dx,4) ;
                printf("\nnew soln x= %5.2f  %6.3f  %5.3f  %5.2f
                                %4.1f\n",x[0],x[1],x[2],x[3],x[4]);

                /* call incident detection */
                n_detect=0;
                n_alarm=0;
                for (j=0;j<123;j++)

                        incl(occ,j,x,&n_detect,&n_alarm);

                dr=((float)n_detect)/69.0;
                far=((float)n_alarm)/8019.0*100.0;
                printf("n_detect=%2d  DR=%5.3f  n_alarm=%3d
                                FAR=%5.3f%\n",n_detect,dr,n_alarm,far);
```

```
                if (far>=old_far || dr<min_dr)
                        {
                        /* forget about current solution x */
                        copy (th,x,4);

                        /* check reduce step size */
                        n_try=n_try+1;
                        if (n-try>max_try)
                                {
                                step_size=step_size/step_div;
                                n_try=0;

                        printf("fail to satisfy min_dr of %4.2f
                                or improve from FAR of %6.4f\n",
                                min_dr,old_far);
                        printf("n_try=%ld  step_size=%4.2f\n",
                                n_try,step_size);

                else

                        /* false alarm rate has been reduced */
                        i=i+1;
                        old_far=far;
                        copy(x,th,4) ;
                        n_try=0;
                        printf("better FAR now i=%2d\n",i);


        printf ("\nprocess terminated because ");
        if (i>=max_i)
                { printf("max Iteration reached\n");}
        else if (old_far<=min_far)
                { printf("minimum FAR achieved\n");}
        else
                { printf("step size is too small\n");}

else
        {
        printf("initial CR > min DR\n");
        printf("change min DR or thresholds and then retry\n");
        }
```

```
/* read upstream/downstream detector data from training.dat */
void  read-data(occ)
        float *occ;
        !
        FILE  *fin;
        int i,j,k,n,time1,time2;
        float line1[16],line2[16];
        float occ_up,occ_dn,occ_dn_pre,occdf,occrdf,docctd,docc;
        fin=fopen("training.dat","r");
        for(i=0;i<69;i++)

                for(j=0;j<68;j++)

                        if (j==0)
                                {
                                fscanf(fin,"%d",&time2);
                                for(k=0;k<16;k++)
                                        {
                                        fscanf(fin,"%f",&line2[k]);

                                occ_up=(line2[2]+line2[1])/2.0;
                                occ_dn=(line2[8]+line2[7])/2.0;
                                occ_dn_pre=(line2[6]+line2[5])/2.0;
                                }
                        else

                                time1=time2;
                                for(k=0;k<16;k++)
                                        i
                                        line1[k]=line2[k];

                                fscanf(fin,"%d",&time2);
                                for(k=0;k<16;k++)

                                        fscanf(fin,"%f",&line2[k]);

                                occ_up=(line1[2]+line2[2])/2.0;
                                occ_dn=(line1[8]+line2[8])/2.0;
                                occ_dn_pre=(line1[6]+line2[6])/2.0;
                                }
                        occdf=occ_up-occ_dn:
                        docc=occ_ dn ;
                        occrdf=occdf/occ_up;
                        docctd=(occ_dn_pre-occ_dn)/occ_dn_pre;

                        *(occ+i*68*4+j*4)=occdf;
                        *(occ+i*68*4+j*4+1)=docctd;
                        *(occ+i*68*4+j*4+2)=occrdf;
                        *(occ+i*68*4+j*4+3)=docc;

                }
        fclose(fin);
```

```
/* read initial solution & parameters */
void read_ini_soln(th,scale,pmin_dr,pmin_far,pstep_size,pmin_step,pstep_div,
                   pmax_try,pmax_i,pseed)
        float *th,*scale;
        float *pmin_dr,*pmin_far,*pstep_size,*pmin_step,*pstep_div;
        rnt *pmax_try,*pmax_i;
        long *pseed;

        FILE *fin;
        int i;
        fin=fopen("ini_soln.dat", "r") ;
        for (i=0;i<5;i++)
                { fscanf(fin,"%f", (th+i)); }
        for (i=0;i<5;i++)
                { fscanf(fin,"%f", (scale+i)); }
        fscanf(fin, "%f %f",pmin_dr,pmin_far);
        fscanf(fin,"%f %f %f",pstep_size,pmin_step,pstep_div);
        fscanf(fin,"%d %d",pmax_try,pmax_i);
        fscanf(fin, "%ld",pseed) ;
        fclose(fin);


/* Roll Wave Supression Algorithm */
int roll_wave(occdf,docctd,occrdf,docc,th,state)
        rnt state;
        float occdf,docctd,occrdf,docc,*th;
        {
        float t1,t2,t3,t4,t5;
        int i;
        t1=*th; t2=*(th+1); t3=*(th+2); t4=*(th+3); t5=*(th+4);
        if (state>=6)

                if (occrdf>=t3)
                        {
                        if (state>=7)
                                { return(E); }
                        else
                                { return(7); }
                        }
                else

                        if (docctd<t2 && docc>=t5 && state<7)
                                { return(1); }
                        else
                                { return(0); }
                        }
```

```
else
        {
        if (docc>=t5)

                if (docctd>=t2)
                        {
                        if (state<1 || state>=5)
                                { return(0); }
                        else
                                { return(state+1); }

                else
                        { return(l); }
                }
        else
                {
                if (state>=1)

                        if (state>=5)
                                { return(0); !
                        eise
                                { return(state+1); }

                else
                        {
                        if (occdf>=t1 && occrdf>=t3 && docc<t4)
                                { return(6); }
                        else
                                }, return(0); }
```

```
/* perform Incident detect-on run for nth case */
void inci(occ,i,th,pn_detect,pn_alarm)
        float *occ,*th;
        int i,*pn_detect,*pn_alarm;
        {
        int j,k,state,detect,alarm;
        float occdf,docctd,occrdf,docc;
        state=0;
        alarm=0;
        detect=0;

        for (j=0; j<68; j++)

                occdf=*(occ+i*68*4+j*4);
                docctd=*(occ+i*68*4+j*4+1);
                occrdf=*(occ+i*68*4+j*4+2);
                docc=*(occ+i*68*4+j*4+3);

                k=roll_wave(occdf,docctd,occrdf,docc,th,state);
                state=k;

                if (state==7)

                        if (i<69 && j>=4 && j<=8)
                                { detect=1; }
                        else if (i<69 && (j<4 || j>8))
                                { alarm=alarm+1; } /*FAR2*/
                        else if (i>=69)
                                { alarm=alarm+1; }


        *pn_detect=*pn_detect+detect;
        *pn_alarm=*pn_alarm+alarm;
        }
```

```
/* get a standard normal random number */
float normal(u1,u2)
        float u1,u2;
        {
        float x,mean,var;
        mean=0.0;
        var=1.0;
        x=mean+sqrt(-var*2.0*log(u1))*sin(2*phi*u2);
        return(x);
        }

/* normalized vector x to a unit vector */
void normalize(dx,n)
        float *dx;
        int n;

        int i;
        float ssq;
        ssq=0.0;
        for (i=0;i<n;i++)

                ssq=ssq+(*(dx+i)) * (*(dx+i));

        for (i=0; i<n;i++)

                *(dx+i)=*(dx+i)/sqrt(ssq);



/* add vectors dx to x */
void add(x,dx,n)
        float *x, *dx;
        int n;

        int i;
        for (i=0; i<n; i++)

                *(x+i)=*(x+i)+*(dx+i);



/* copy vectors x to y */
void copy(x,y,n)
        float *x,*y;
        int n;

        int i;
        for (i=0;i<n;i++)

                *(y+i)=*(x+i);
```

```
/* get an array x in which each element has a */
/* std normal distribution */
void get_dx(dx,seed,n)
        float *dx;
        long seed:
        int n;
        {
        int i;
        float a,b;
        for (i=0;i<n;i++)
                {
                a=rand(&seed);
                b=rand(&seed);
                *(dx+i)=normal(a,b);



/* scale x according a scaling vector */
void scale_dx(dx,scale,step_size,n)
        float *dx,*scale,step_size;
        int n;
        {
        int i;
        for (i=0;i<n;i++)
                {
                *(dx+i)=*(dx+i)*(*(scale+i))*step_size;

        }


/* get uniformly distributed random number */
float rand(px)
        long *px;

        float v;
        long ki;
        ki=*px/127773;
        *px=16807*(*px-ki*127773) -ki*2836;
        if(*px<0)
                { *px=*px + 2147483647; }
        v=*px*4.656612875e-10;
        return(v) ;
        }
```