

# UC Berkeley

## Earlier Faculty Research

### Title

A Neuro-Genetic-Based Universally Transferable Freeway Incident Detection Framework

### Permalink

<https://escholarship.org/uc/item/3q93f0jp>

### Author

Abdulhai, Baher

### Publication Date

1996

UNIVERSITY OF CALIFORNIA,  
IRVINE

A Neuro-Genetic-Based Universally Transferable Freeway Incident Detection  
Framework

DISSERTATION

submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in Civil Engineering

by

Baher Abdulhai

Dissertation Committee:

Professor Stephen G. Ritchie

Professor Wilfred W. Recker

Professor R. Jayakrishnan

1996

© Baher Abdulhai, 1996  
All rights reserved

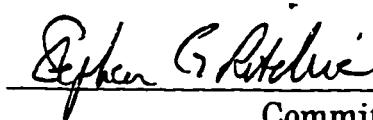
The dissertation of Baher Abdulhai is approved  
and is acceptable in quality and form  
for publication on microfilm



---



---



---

Committee Chair

University of California, Irvine

1996

## DEDICATION

TRULY  
MY PRAYER,  
AND MY SERVICE OF SACRIFICE,  
MY LIFE AND MY DEATH  
ARE ALL FOR ALLAH  
THE CHERISHER OF THE WORLDS.  
NO PARTNER HAS HE:  
THIS AM I COMMANDED,  
AND I AM THE FIRST  
OF THOSE WHO SUBMIT TO HIS WILL (MUSLIMS)

# TABLE OF CONTENTS

LIST OF FIGURES.....	IX
LIST OF TABLES .....	XI
ACKNOWLEDGEMENT.....	XXII
CURRICULUM VITAE.....	XXII

## CHAPTER 1

### INTRODUCTION

1.1 BACKGROUND.....	1
1.2 RESEARCH OBJECTIVE .....	7
1.3 DISSERTATION ORGANIZATION .....	7

## CHAPTER 2

### UNIVERSALITY REQUIREMENTS AND EXISTING AID ALGORITHMS

2.1 INTRODUCTION.....	9
2.2 UNIVERSALITY REQUIREMENTS .....	10
2.3 EXISTING AID ALGORITHMS .....	15
2.3.1 <i>Neural Network Models and the MLF-Based AID Algorithm</i> .....	15
2.3.2 <i>The California Algorithms</i> .....	18

2.3.3 <i>The Minnesota Algorithm</i> .....	20
2.4 LACK OF UNIVERSALITY OF EXISTING ALGORITHMS .....	22
2.5 THE NEED FOR A UNIVERSAL FRAMEWORK.....	24

**CHAPTER 3**

**FREEWAY STUDY SITES, INCIDENT AND LOOP DETECTOR DATABASES**

3.1 INTRODUCTION.....	26
3.2 TEST SITE SELECTION .....	26
3.2.1 <i>The I-880 Freeway Test Site</i> .....	26
3.2.3 <i>The I- 35W Freeway Test Site</i> .....	27
3.2.3 <i>The SR-91 Freeway Test Site</i> .....	29
3.3 TRAFFIC EVENT DATABASES DEVELOPMENT.....	32
3.3.1 <i>The I-880 Database</i> .....	32
3.3.2 <i>The I-35W Database</i> .....	35
3.3.3 <i>The SR-91 Database</i> .....	37
3.4 CONCLUDING REMARK .....	38

**CHAPTER 4**

**DETECTABLE INCIDENTS AND ACCEPTABLE LIMITS ON DETECTION PERFORMANCE**

4.1 INTRODUCTION.....	40
4.2 WHAT IS AN INCIDENT .....	41
4.3 DETECTABLE CLASSES OF INCIDENTS .....	41
4.4 PERFORMANCE MEASURES .....	45

4.5 TMC-ACCEPTABLE LIMITS ON DETECTION PERFORMANCE.....	46
---	----

## **CHAPTER 5**

### **INVESTIGATING ADVANCED NEURAL NETWORKS FOR IMPROVED INCIDENT DETECTION**

5.1 INTRODUCTION.....	50
5.2 THE LOGICON PROJECTION NETWORK™.....	51
5.2.1 <i>Theory</i> .....	51
5.2.2 <i>Advantages of the Logicon Network</i> .....	56
5.3 THE PROBABILISTIC NEURAL NETWORK (PNN).....	57
5.3.1 <i>Theory: Multivariate Bayesian Discrimination and the PNN</i> .....	58
5.3.2 <i>Advantages of the PNN-family of networks</i> .....	64
5.3.3 <i>Statistical Distance and The Proposed Modified PNN2</i> .....	66
5.4 PRELIMINARY ANALYSIS OF THE LOGICON AND THE PNN NETWORKS FOR INCIDENT DETECTION .....	73
5.4.1 <i>Training the Logicon Network</i> .....	74
5.4.2 <i>Training the PNN Networks</i> .....	80
5.4.3 <i>Incident Detection Performance</i> .....	82
5.4.4 <i>Comparative Evaluation and Preliminary Conclusions</i> .....	85

## **CHAPTER 6**

### **DETAILED DEVELOPMENT OF THE PNN-BASED FRAMEWORK**

6.1 INTRODUCTION.....	89
-----------------------	----



6.2	TRANSFERABILITY TESTING AND EVALUATION .....	90
6.3	IN-SERVICE PERFORMANCE IMPROVEMENT OF THE PNN.....	94
6.3.1	<i>On-Site Real-Time Retraining</i> .....	94
6.3.2	<i>Discussion</i> .....	102
6.4	ENHANCED FEATURE EXTRACTION FOR UNIVERSAL TRANSFERABILITY .....	104
6.4.1	<i>Causes of the Lack of Transferability Problem</i> .....	104
6.4.2	<i>Enhanced Feature Extraction</i> .....	110
6.4.3	<i>Implementation of the New Features</i> .....	117
6.5	POST-PROCESSOR INTERPRETER OF AID OUTPUT .....	122
6.5.1	<i>Drawbacks of the Isolated Binary Outputs</i> .....	126
6.5.2	<i>Continuous Updating of Incident Probabilities</i> .....	130
6.5.3	<i>Alternative Updating Scheme of Incident Probabilities</i> .....	137
6.6	THE FULL UNIVERSAL SYSTEM.....	138

## **CHAPTER 7**

### **AN IMPROVED GENETICALLY OPTIMIZED MULTI- $\sigma$ PNN**

7.1	INTRODUCTION.....	140
7.2	A MULTI- $\sigma$ ADAPTIVE PNN.....	141
7.2.1	<i>Conventional Adaptation Methods</i> .....	142
7.3	GENETIC ALGORITHMS.....	145
7.3.1	<i>Introduction</i> .....	145
7.3.2	<i>Genetic Algorithms: General Uses and Advantages</i> .....	148
7.3.3	<i>Genetic Synthesis and Optimization of Neural Networks</i> .....	149
7.4	IMPROVED GAPNN-BASED INCIDENT DETECTION .....	150

7.4.1 *Training and Testing of the GAPNN*..... 151  
7.4.2 *Comparative Evaluation and Concluding Remarks*..... 156

**CHAPTER 8**

**SUMMARY, CONCLUSIONS AND RECOMMENDATIONS**

8.1 SUMMARY..... 159  
8.2 CONCLUSIONS..... 161  
8.3 RECOMMENDATIONS..... 161

**REFERENCES..... 163**

## LIST OF FIGURES

Figure 2.1	The MLF neural network .....	17
Figure 2.2	The California algorithm # 8 .....	19
Figure 3.1	The I-880 Freeway Site .....	28
Figure 3.2	The I-35W Freeway Site.....	30
Figure 3.3	The SR-91 Freeway Site.....	31
Figure 4.1	Survey Form Mailed to Different TMCs. ....	49
Figure 5.1	The Logicon Projection Network™ .....	53
Figure 5.2	Stereographic projection in the Logicon network .....	54
Figure 5.3	The Original Probabilistic Neural Network (PNN) .....	62
Figure 5.4	Improper Use of Euclidean Distance.....	67
Figure 5.5	Axes Rotation for Statistical Distance .....	69
Figure 5.6	The Modified Probabilistic Neural Network (PNN2).....	72
Figure 5.7	Logicon, PNN, PNN2 and MLF Performance on data set 3.....	84
Figure 6.1	Performance Envelopes Before and After Real-Time Updating on the I-880 Patterns.....	100
Figure 6.2	Performance Envelopes Before and After Real-Time Updating on the I-35W Patterns. ....	101
Figure 6.3.a	Incident Data in Two Dimensional Input Space.....	105
Figure 6.3.b	Normal Data in Two Dimensional Input Space .....	105
Figure 6.4	Varying Pre-incident Conditions and Post-incident Effects Contribute to Classes Overlap. ....	109
Figure 6.5	More Distinct Incident and Normal Patterns Using the Proposed Features .....	115
Figure 6.6	Performance Envelopes after Transferability to the I-880 Site Using Conventional and New Features.....	121

Figure 6.7	Performance Envelopes after Transferability to the I-35W Site Using Conventional and New Features.....	124
Figure 6.8	Updated Incident Probabilities vs. Time for Incident #1 .....	135
Figure 6.9	Updated Incident Probabilities vs. Time for Incident #2 .....	136

## LIST OF TABLES

Table 2.1	Lack of Universality of the MLF, CA, and Minnesota Algorithms .....	25
Table 4.1	DR and FAR Limits Extracted from TMC Responses. ....	48
Table 5.1	Testing the Logicon, PNN, PNN2 and the MLF on Data Set 3. ....	83
Table 5.2	Universality Comparison of the Logicon, PNN2 and the MLF .....	86
Table 6.1	Testing Results on the I-880 Data Set.....	92
Table 6.2	Testing Results on the I35W Data Set. ....	93
Table 6.3	Performance Improvements After Real-Time Updating on the I-880 Patterns. ....	98
Table 6.4	Performance Improvements After Real-Time Updating of the I35W Patterns. ....	99
Table 6.5	Examples of the New Input Features (Occupancy Values Example).113	
Table 6.6	Transferability to the I-880 Site Using Conventional and New Features. ....	120
Table 6.7	Transferability to the I-35W Site Using Conventional and New Features. ....	123
Table 6.10	Universality of the PNN-based framework.....	139
Table 7.1	Performance of the PNN and GAPNN, trained on SR-91 data and tested on I-880 data.....	153
Table 7.2	Performance of the PNN and GAPNN, trained on SR-91 data and tested on I-35W data. ....	154
Table 7.3	Performance of the PNN and GAPNN, trained on partial I-880 data and tested on I-880 data.....	155
Table 7.4	Performance of the PNN and GAPNN, trained on partial I-35 data and tested on I-35W data. ....	156

## ACKNOWLEDGEMENT

I would like to express my deepest appreciation and gratitude to Professor Stephen Ritchie, my advisor and my committee chair, for his guidance and support through out my studies. I thank Professor Wilfred Recker for being always there whenever I needed his advice and support. I thank Professor M. McNally and Professor R. Jayakrishnan for their invaluable contributions to my knowledge. I thank Professor Medhat Haroun for his encouragement and support. I thank them all for creating such a powerful and enjoyable graduate program, and for giving me the opportunity to be a member of the UCI-ITS team. Credit is also due to my colleagues who are the team.

I thank the University of California Transportation Center for granting me their prestigious UCTC Dissertation Fellowship and research grant twice. I thank the University of California Irvine for granting me the U.C. Regents Fellowships.

I thank my family, my Dad, Mom, my wife Nayera, and my brother for their continuous support and encouragement and for believing in me.

This research is supported by the Partners for Advanced Transit and Highways (PATH), the University of California Transportation Center (UCTC), and the California Department of Transportation (Caltrans) Advanced Traffic Management System (ATMS) Testbed research program.

## **CURRICULUM VITAE**

**Baher Abdulhai**

- 1988            B.Sc. in Civil Engineering (Honors), Cairo University, Egypt.
- 1988-1991     Assistant Teacher, Department of Civil Engineering, Cairo University, Egypt.
- 1991            M.Sc. in Civil Engineering (Highway, Pavement and Airport Engineering) , Cairo University, Egypt.
- Thesis: "Application of Expert Systems to Pavement Maintenance Management".
- Professor Essam Sharaf, Chair
- 1991- pres.    Associate Teacher, Department of Civil Engineering, Cairo University, Egypt.
- 1991-1993     Graduate Student, Department of Civil Engineering, University of Alberta. Canada.
- 1993-1996     Graduate Student Researcher, Institute of Transportation Studies and Department of Civil and Environmental Engineering, University of California, Irvine.
- 1996            Ph.D. in Civil Engineering (Transportation), University of California, Irvine.
- Dissertation: "A Neuro-Genetic-Based Universally Transferable Freeway Incident Detection Framework."
- Professor Stephen G. Ritchie, Chair

**FIELD OF STUDY:**

**Transportation Engineering**

## **ABSTRACT OF THE DISSERTATION**

### **A Neuro-Genetic-Based Universally Transferable Freeway Incident Detection Framework**

by

**Baher Abdulhai**

Doctor of Philosophy in Engineering  
University of California, Irvine, 1996  
Professor Stephen G. Ritchie, Chair

A universal freeway incident detection framework is a task that remains unfulfilled despite the promising approaches that have been recently explored. The need for an operationally successful incident detection and management system as a vital component of any advanced traffic management system, is well established and recognized. Only recently however, researchers and practitioners have begun to increasingly realize that for an incident detection framework to be universally operational and successful, it needs to fulfill all components of a set of recognized needs. It is the objective of this research to define those universality requirements and produce an incident detection framework that possesses the potential to fulfill them.

A new potentially universal freeway incident detection framework has been proposed, developed and evaluated. The research effort was started by defining a comprehensive set of requirements that any universal incident



detection algorithm or framework should fulfill. Among these requirements, an incident detection algorithm needs to be operationally accurate, automatically transferable, and capable of automatically adapting to changes in the freeway environment. This set of universality requirements was used as a template against which all algorithms within the scope of this study have been evaluated. Three major incident and loop detector databases were heavily utilized, two of which are unprecedented real databases collected from two major freeway sites in California and Minnesota, namely the Alameda County's I-880 freeway database and the Minneapolis' I-35W database. The universality of the most well known existing incident detection algorithms was tested using the above databases. Serious lack of universality, particularly transferability, was detected in all existing algorithms. Prior to the development of the new universal framework, limits on acceptable performance were elicited from TMC surveys conducted as part of this effort. Preliminary investigation of two promising advanced neural networks, namely the LOGICON and the PNN, was conducted. The PNN was more appealing due to its universality potential. The PNN was modified using a principal components transformation layer that resulted in performance enhancements. This together with its potential universality, led to the choice of the modified PNN for in-depth development. The in-depth development stage was divided into three phases. The first was the extraction of a new and improved input feature set that produced more distinct classes in the input feature space. The new features enhanced the transferability of the PNN and made the framework more compliant with the universality requirements. The second phase was the on-site real time retraining of the PNN after transferability, a phase that produced near optimal classification results and detection performance. The third phase was the development of a post processor output interpreter that linked the isolated 30 second outputs of the PNN and produced a sequentially updated

probabilistic measure of existence of an incident in the field. The overall PNN-based framework was found to be fully compliant with the entire set of universality requirements. Finally a new approach for training a multi-smoothing-parameter version of the PNN was investigated. The approach utilized genetic algorithms for optimizing the selection of the smoothing parameters. Obtained results indicated an improvement in performance over the single smoothing parameter PNN but at the expense of longer training time.

The superiority and universality of a particular advanced neural network model, namely the PNN, was concluded in this research, as compared to the Logicon and the MLF neural networks, as well as existing conventional freeway incident detection algorithms. Adding the principal components transformation layer to the PNN was found to enhance its performance. Although the genetically optimized version of the PNN showed better transferability, both versions showed equally good performance after retraining. The PNN was concluded to be more practical for TMC implementation due to its instantaneous training capabilities.

# CHAPTER 1

## INTRODUCTION

### 1.1 BACKGROUND

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 US attributable to non-recurring congestion range up to about 60% (Lindly, 1987).

Successful automated detection of such 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. An effective incident detection and management system is now a well recognized key component of any successful Advanced Traffic Management System (ATMS), the heart of any Intelligent Transportation System (ITS) in general.

Unfortunately, the performance of published conventional approaches to automatic incident detection has proven inadequate for every day use at Traffic Management Centers (TMCs). Such inadequacy is believed to stem from three main sources: the *first* is the less-than-perfect performance at the original site that the algorithm was developed for, while the *second* is the lack of transferability to any other new site, and the *third* is the absence of important issues such as the prior (predicted) probability of occurrence of incidents, posterior probability of an incident after an alarm, and the unequal costs of misclassifying a traffic pattern, amongst other issues that will be discussed in due course. Less than perfect performance at any site is characterized by unacceptably high False Alarm Rate (FAR) and/or low Detection Rate (DR) and the conflict between the two, as well as unacceptably long mean Time to Detection (TTD). At the original site, this is often caused by lack of sufficiently representative real loop and incident data, modest choice of input features to feed the algorithm, and modest pattern recognition capabilities of the algorithms, on theoretical grounds or due to poor training. By lack of transferability is meant that when an algorithm is used in an environment that differs significantly from the one it was developed for, without retraining, and the performance drops significantly. This is a problem that calls for re-development or re-training of the algorithm for every new environment. A new environment could be defined as any freeway site that has significantly different statistics. Site statistics change,

of course, from site to site, and less evidently at the same site with long spans of time, and may cause the existing knowledge content of the previously trained system to become less relevant over time.

More advanced algorithms have been recently proposed and developed by different researchers using approaches such as neural networks (Cheu and Ritchie, 1995), filtering techniques (Stephanedes and Hourdakis, 1996) and catastrophe theory (Hall et al 1991). The neural network approach has shown the highest potential for detecting lane blocking incidents. However, all algorithms still lack several of the attributes that makes an incident detection system universally acceptable. Even a comprehensive list of such universality requirements is not readily available. Very recent research *plans* have been made (Payne and Thompson 1996) to consolidate the benefits of the different promising incident detection approaches in a modular structure that would benefit from them all. However there is no one single, stand alone algorithm that has shown any promise to fulfill all the expectations simultaneously, and hence no one single algorithm, to date, is universal.

Such lack of universality leads to operational failure that is manifested in the lack of operator confidence in the detection system, wasted resources verifying alarms that are not real incidents, and delayed or no response to real incidents that remain undetected. The need for improved systems

remains pressing, particularly with the advent and progress of intelligent transportation system (ITS) concepts for integrated freeway and arterial networks. These will rely heavily on the ability to automatically detect non-recurring traffic congestion in order to adjust traffic management plans in real-time for optimal control and provision of traveler information systems. More effective incident detection methods should result in responsive emergency systems and traffic management. Ultimately this would result in reduced motorist delay and expense, increased convenience, and increased fulfillment of the prime objective of modern transportation.

It is the focus of this research to clearly define one comprehensive set of universality requirements for automated incident detection systems, and produce an improved framework that fulfills the defined requirements. The proposed framework will have a core algorithm based on neural networks and genetic algorithms.

Artificial neural networks (ANNs) -from the field of Artificial Intelligence- are newly emerging, rapidly advancing, parallel distributed information processing structures based on a simplified model of the functioning of the human brain. Neural networks consist of many processing elements (PEs) interconnected with each other. The neurons are organized in groups, one group receives external inputs, another group communicates the output to some external sources, and the rest of the neurons are dedicated for internal

representation and processing of the input patterns and mapping them to the output, and hence called hidden neurons. Each PE in the network receives signals from other PEs, weighted by the interconnection weights, and transmits the processed signal to other PEs. Information is thus represented and processed in a parallel and distributed fashion across the network. The major advantages of neural networks are: fast processing speed, parallel and distributed representation and processing of information, the ability to be trained to perform non-linear mappings of patterns, and the ability to produce good classification results with imperfect input data (Hecht-Nielson, 1990, Simpson, 1990). They are capable of replicating rule-based-like behavior while simultaneously considering all the contributing factors and constraints. They do not rely on assuming specific statistical distributions. They learn from examples. They are fault tolerant and degrade gracefully. Most importantly, they outperform conventional pattern recognition approaches, at least for the purpose of this research.

Genetic Algorithms - also from the field of AI - are inspired by evolution in the real world, which is controlled by the process of natural selection. Organisms most suited for their environment tend to live long enough to reproduce, whereas less-suited organisms often die before producing children or produce fewer and/or weaker children with less and less chances of survival. Genetic algorithms (GA) can be defined as a problem-solving method that uses genetics as its model for problem solving. They apply the

rules of reproduction, gene crossover, and mutation to a population of candidate solutions or pseudo-organisms so that those organisms can pass beneficial and survival-enhancing traits to new generations (Chambers 1996). The GA approach is a very powerful optimization method capable of efficiently searching through a large solution space without getting stuck in local minima, and does not require feedback links for training. Each candidate solution to a problem is considered a point in the possible solution space, or a chromosome. The first step in finding the optimum value is to select a population of chromosomes at random, and compute the fitness function (detection accuracy) for each of these points. The search proceeds by picking two points at random, biasing the selection process toward those with higher scores using a roulette-wheel-like mechanism. At this juncture, one or both of two operations take place. The first is to create a third point by randomly combining various parts of the two points which is called reproduction by crossover. Alternatively, one of the two points can be selected, and a certain number of its components randomly perturbed, which is called mutation. This is equivalent to taking a small step in a random direction from the initial point searching locally for a solution. After a new point is generated by either process or a combination of processes, the new point is evaluated. The search process proceeds until a pre-determined stopping condition is reached. GA are used in this research to optimize neural network parameters.



## **1.2 RESEARCH OBJECTIVE.**

The purpose of this research is to develop an advanced universal freeway incident detection framework, utilizing state of the art neural networks and genetic algorithms. The research is focused on enhancing the transferability of AID algorithms by improving the input features on statistical and traffic flow grounds, and by using modern ANN paradigms that are capable of adapting to new operating environments. Ties are established between these new technologies and existing statistical theory in order to lighten the way to advancements. Newly available, unprecedented real-world traffic and incident databases collected from several major freeways in the U.S. are heavily utilized in the development and testing of the proposed advanced algorithms.

## **1.3 DISSERTATION ORGANIZATION**

The dissertation consists of eight chapters, and is organized as follows:

Chapter 1 introduces the background to the problem and the objective of this research.

Chapter 2 introduces a set of requirement that any incident detection system should fulfill in order to be universal. Existing algorithms are reviewed and

evaluated against these requirements. The need for a new framework is established.

Chapter 3 describes the freeway sites and the associated databases that are used for the development and evaluation of the new system as well as evaluating the existing ones.

Chapter 4 defines detectable incidents, the performance measures used in the study, and draws the acceptable limits on these measures from traffic management center surveys.

Chapter 5 examines promising advanced neural network architectures, the possible modifications to improve them and selects the most promising architecture for further in-depth development and investigation.

Chapter 6 describes the detailed development, testing and evaluation of the proposed universal incident detection framework.

Chapter 7 introduces genetic algorithms and describes their use for training an enhanced version of the system.

Chapter 8 summarizes the research, and presents the conclusions and the recommendation for future research.

## **CHAPTER 2**

### **UNIVERSALITY REQUIREMENTS AND EXISTING AID ALGORITHMS**

#### **2.1 INTRODUCTION**

This chapter defines one possible set of universality requirements for Automated Incident Detection (AID) algorithms. These requirements are used throughout this research. They represent goals for the new framework to achieve, and an evaluation template for existing or even future AID algorithms. The assumption is the more of these requirements any AID algorithm fulfills the closer it is to being universal. Three existing AID algorithms are reviewed in this chapter, namely, the MLF neural network algorithm, the Minnesota filtering algorithm, and the California algorithm. These algorithms are used in a comparative evaluation with the proposed framework. From among the existing body of AID algorithms, the first is chosen because of its recently demonstrated superiority (Cheu 1994); the second is chosen because it uses a modern filtering approach; and the third because of its wide spread use in the field which makes it a benchmark in almost every study. Which of the universality attributes each of these existing algorithms possess is then discussed. Absence of one or more universality attributes is considered a limitation that the proposed framework is designed to overcome.

## 2.2 UNIVERSALITY REQUIREMENTS

Putting together TMC survey results, theoretical reasoning, intuition, and experience, a set of universality attributes and capabilities is compiled. For an incident detection algorithm to be universal, it needs to possess the following:

1. **High performance:** in terms of high Detection Rate (DR%), low False Alarm Rate (FAR%) and short mean Time To Detection (TTD). High performance could only be achieved through careful selection of distinctive input features, a powerful pattern recognition logic, and training data that represents all possible scenarios. Acceptable limits on the above performance indicators or variables (DR, FAR, and TTD) have been elicited from a variety of Traffic Management Centers (TMCs) as will be detailed later.
2. **Fast, robust and automated training and retraining:** training of the algorithm should neither be complex nor tediously time consuming. In fact, there are no known limits on what a reasonable training time would be. However, the faster the training/retraining processes the better the algorithm, and the higher the potential of its on-site real-time adaptability, all other attributes being equal. Equally important, training should not be dependent on the hand-crafting skills of the developer. Such dependence necessitates an attended retraining after transferability

and/or significant changes in site statistics, an obstacle that hinders the on-site real-time updating of the knowledge content of the AID algorithm.

3. **Reasonable TMC implementation requirements:** no recalibration requirements that require skilled system developers in a TMC is one example previously discussed. No special site-dependent detector placement configuration is another. Successful implementation of the algorithm should not require special detector placement configuration in the field. For instance, some algorithms require a special loop detector placement configuration within a funneling freeway section (Wei-Hua Lin, 1995). Such a requirement would limit the applicability of the algorithm in the absence of the required configuration.
4. **Transferable logic:** the logic or theory on the basis of which the algorithm is built should not be limited to certain operational circumstances.
5. **Reasonably transferable training/calibration parameters:** an algorithm trained on one site should be usable in other new environments with as little performance deterioration as possible. A new environment could be defined as any freeway site that has significantly different statistics. Site statistics change, of course from site to site, and less evidently at the same site as it undergoes significant changes with long spans of time. The less the performance deterioration after transferability,

the less the burden on any subsequent on-site recalibration/updating processes.

6. **Minimal initial training data requirements:** as is well known, real incident data are not only very sparse but also very difficult to obtain. It would be very time consuming, if not impractical to try to collect real incident data that are diverse enough to represent all possible scenarios, and accurate enough to result in minimal calibration errors. It is easy to realize that a serious limitation of most existing incident detection algorithms is the quality of initial training due to the nature of the data used for calibration. They are often calibrated using either a very limited and/or inaccurate set of "real" incident data or a large set of artificially simulated data with all the inherent simulation problems such as, for instance, the ability of the embedded car-following or lane-changing models to replicate actual driver behavior. This may limit the quality of the resulting algorithms and perhaps also limit their transferability. Hence, a successful algorithm should be capable of being up and running using minimal incident data, and then automatically improve with time in service as more usable data become available.
7. **Account for prior probabilities of incidents:** the algorithm should incorporate into an incident alarm decision the predicted prior probability

of occurrence of an incident based on such factors as weather condition, traffic conditions, road surface conditions, and freeway geometry.

**8. Account for the unequal costs of misclassifying traffic patterns:**

Although a false alarm and a missed incident are both misclassifications, their costs are not equal. One might be more costly than the other depending on several factors a few of which might be the location of the freeway section, the time of the day, TMC size, capabilities and preferences.

**9. Capable of producing the posterior probability of an incident:**

there usually are varying extents of uncertainty associated with AID algorithm-generated alarms. The certainty of an alarm depends among other things on the prior probability of an incident, and the number of preceding alarms (which may be correlated to the elapsed time since the onset of the incident and extent of closure and queue formation on the freeway). The algorithm should be capable of producing a probabilistic estimate of the certainty associated with an alarm, and update this probability in a continuous fashion during the incident event.

**10. Estimate incident severity:** a key requirement that helps prioritize the allocation of TMC resources in response to several simultaneous incident alarms.

11. **Capture incident duration:** a key information required in order indicate the restoration of normal traffic flow.
12. **Statistical and theoretical soundness and clarity:** key factors that underlie its acceptability. The more consistent the calibration process is and the fewer the number of heuristics involved, the more general the algorithm would be.
13. **Immunity to minor traffic fluctuations:** the algorithm should not be affected by minor, and very short term traffic fluctuations which tend to cause false alarms.
14. **Immunity to bottleneck effects:** the algorithm should be capable of distinguishing between true incident patterns and incident-like patterns due to physical bottlenecks that cause queuing patterns similar to an incident. It should also be immune to virtual bottleneck effects caused by abrupt geometry changes or demand changes at major on and off ramp locations. The latter scenario may be called a virtual bottleneck because of the associated sudden change in the traffic flow variables from the upstream loop station to the downstream loop station of the freeway section, which is similar to the effect of an incident and the effect of physical bottlenecks, despite the absence of both.
15. **Immunity to consistent loop detector biases:** the algorithm should not be affected by consistent loop biases such as an upstream station



giving consistently higher occupancy readings than the downstream station for no obvious reasons. Such a situation has been observed in field data.

The above template of requirements and attributes, if fulfilled, could yield a complete, operationally successful incident detection algorithm. The presence or absence of these attributes will be used to evaluate the proposed framework as well as any other algorithm that might be used for comparative purposes.

## **2.3 EXISTING AID ALGORITHMS**

### **2.3.1 Neural Network Models and the MLF-Based AID Algorithm**

Cheu (1994) demonstrated the potential of the artificial neural network approach for automated detection of lane-blocking incidents on freeways. Three types of neural network models, namely the multi-layer feed-forward neural network (MLF), self-organizing feature map (SOFM), and adaptive resonance theory 2 (ART2) were investigated. Each of the models was trained to detect only lane-blocking incidents in any freeway section, bounded by detector stations of up to one mile spacing. Training was performed with data generated from the well known INTRAS microscopic

freeway simulation model, calibrated to the SR-91 freeway study site in Orange County, California.

The inputs to all the neural network models were the direct values of the station average volume and occupancy accumulated over 30 seconds, for up to the past five intervals.

Based on training results, the MLF with both upstream and downstream station input achieved the highest detection and the lowest false alarm rate among all the neural network models. The MLF also showed better performance than the California algorithm #8, the McMaster algorithm and Minnesota algorithm. Similar superiority of the MLF was reported by Cheu (1995) from testing on a limited number of real incidents from different freeway sites.

The topology of the recommended MLF network is shown in Figure 2.1. An input layer, one hidden layer, and a single node output layer were used. Uni-directional connections between layers without within-layer connections were employed. Training of the network was achieved using the Back Propagation method. The number of hidden neurons, the learning parameters, and the transfer function, were all hand-crafted using extensive trial and error runs. No systematic automated approach for optimizing their selection was employed.

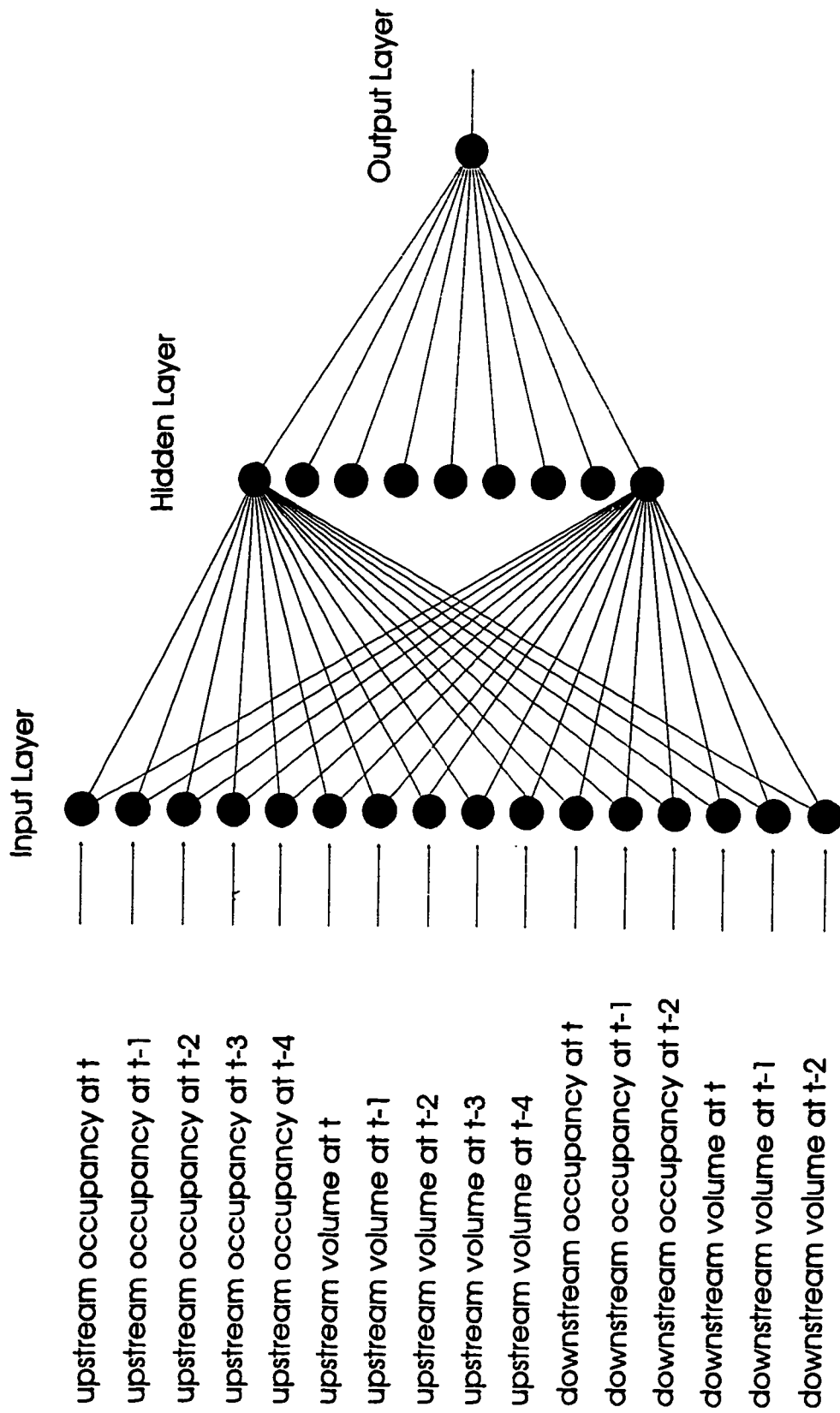


Figure 2.1 The MLF neural network (from Cheu 1994)

### **2.3.2. The California Algorithms**

A family of 11 algorithms developed by Payne et al. (1976) is known as the California algorithms. These algorithms detect discontinuities in occupancy values between the upstream and downstream loop stations of a freeway section. Once a significant discontinuity is detected, an incident alarm is issued. Input features to the algorithms are based on 60 second occupancy averages, and take the form of absolute, relative and temporal differences in occupancies from the two loop stations. These differences are compared against calibrated thresholds. A decision tree is used to classify the traffic condition at the given time interval into different states. The California algorithm has been used by Cheu (1994) and is also used in this research because of its widespread use in TMCs. The California #8 in particular was selected because of its appealing five minute roll-wave suppression logic that aims to reduce false alarms due to shock waves approaching from the downstream. The structure of the decision tree, and the input features are shown in Figure 2.2. Five thresholds were used and calibrated by Cheu (1994) from historical data. Sixty second averages were used at 30 second application intervals. A large number (68,607) threshold sets was first generated using intermediate values within the probable range of each of the five thresholds. For each set of thresholds, the algorithm was applied to simulated incident databases. For each level of detection, the threshold set that gave the lowest false alarms was selected.

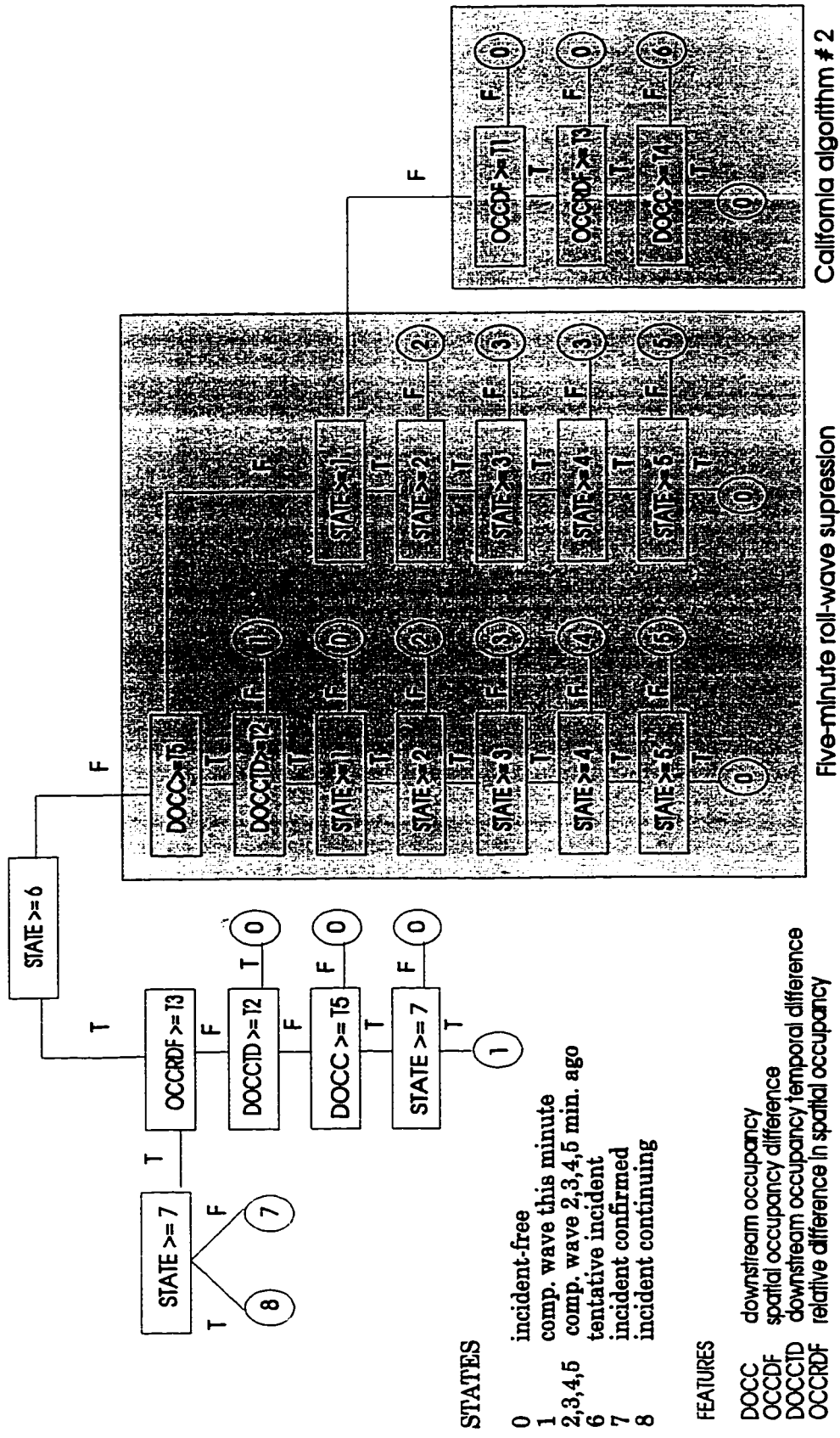


Figure 2.2 The California algorithm # 8 (from Cheu 1995, from Payne et al., 1976)

### 2.3.3. The Minnesota Algorithm

The Minnesota algorithm was developed at the University of Minnesota by Stephanedes and Chassiakos (1993 a & b). The particular version used in this study was calibrated by Cheu (1994). The algorithm operates at 30 second intervals, and uses as input one minute average occupancy across all lanes, at the upstream and downstream stations.

At every 30 second interval (say, at time  $t$ ), the one minute station average occupancy at the upstream station,  $o_t^u$ , and at the downstream station,  $o_t^d$ , are used to compute the spatial occupancy difference ( $o_t^u - o_t^d$ ). The average spatial occupancy differences from  $t-5$  to  $t$ , and from  $t-15$  to  $t-6$  intervals are next computed:

$$y_t^a = \frac{1}{6} \sum_{k=0}^5 (o_{t-k}^u - o_{t-k}^d)$$

$$y_t^b = \frac{1}{10} \sum_{k=6}^{15} (o_{t-k}^u - o_{t-k}^d)$$

The Minnesota algorithm hypothesizes that, when an incident occurs at time  $t-5$ ,  $y_t^a$  will increase and  $y_t^b$  would be greater than  $y_t^a$ . Therefore, this algorithm performs tests on two ratios:

$$RAT1 = \frac{y_t^a}{m_t}$$

$$RAT2 = \frac{y_t^a - y_t^b}{m_t}$$

where  $m_t$  is a normalization factor to account for traffic conditions prior to the incident, defined by:

$$m_t = \frac{1}{10} \max\left(\sum_{k=6}^{15} o_{t-k}^u, \sum_{k=6}^{15} o_{t-k}^d\right)$$

The RAT1 tests the discontinuity in spatial occupancy over the past three minutes (six intervals), while RAT2 evaluates the increase in spatial occupancy difference between the two time windows. An incident warning is declared at time  $t$  if both RAT1 and RAT2 are greater than their respective thresholds, Thr1 and Thr2 respectively. These thresholds have to be calibrated with historical data prior to application.

Cheu (1994) calibrated the algorithm using simulation data. Probable ranges for the two thresholds in the algorithm were first determined from the values reported by Chassiakos and Stephanedes (1993), from which 550 pairs of threshold values were obtained. For each pair of threshold values, the DR and FAR were computed from all the incidents in the database. The initial results showed that these threshold values gave a narrow range of high DRs, and at the same time, very high FARs. The range of each of the two

threshold values was then extended up to twice the largest value reported by Chassiakos and Stephanedes (1993), resulting in a total of 2350 combinations. The calibrated threshold values were selected as described in Chassiakos and Stephanedes (1993). First, the maximum FAR was varied from 0% to 4% at 0.1% increments. For each value of maximum FAR, the threshold set which resulted in the highest DR and yet had a FAR lower than the maximum FAR was selected.

#### **2.4. LACK OF UNIVERSALITY OF EXISTING ALGORITHMS**

In this section the MLF, California, and Minnesota algorithms are evaluated relative to the universality requirements defined earlier and in the same sequence. In a previous study (Cheu 1994), the MLF was shown to outperform both the California and Minnesota algorithms on a common database, and is hence considered to be the best performing algorithm available. However, as will become evident later in this research, none of the algorithms satisfies the performance requirements or expectations of Traffic Management Centers (TMCs). All three algorithms, without exception, require careful hand crafting of the training parameters and thresholds, a process that is very time consuming, requires skill, and lacks automation. Customization of these algorithms by TMC personnel is not possible if they do not have the necessary skills and background. There is no obvious reason



to think that the logic underlying any of the algorithms is not transferable to different sites, subject to availability of new data and a skilled system developer to re-calibrate them. As will also be evident from results obtained in this research none of the already trained/calibrated algorithms is transferable to new sites. All algorithms require extensive data for training. Because none of them has an underlying theory that relies on any distance measure for classification, generalization on the basis of a few training data vectors is not possible. None of the three algorithms is capable of accounting for the varying prior probabilities of incidents in the field or the expected cost of misclassifying an incoming input vector. All algorithms produce isolated binary decisions every time interval, without any reflection on the confidence in the decision or the posterior probability of existence of a certain condition in the field. It is worthy to mention, however, that the MLF's output is first produced in a continuous fashion between 0 and 1, and is then translated to either 0 or 1. Before this translation takes place, the continuous output could be interpreted as a measure of confidence. Nevertheless, this is not an explicit probability measure. Neither the California nor the Minnesota algorithms is capable of reflecting the severity of an incident. However, due to the nature of its continuous output, the MLF can reflect the severity of a traffic blockage as the output gets closer to 1. All the three algorithms keep indicating an incident condition as long as the incident's effect is still in the field, and they all capture the temporal profile of an incident's formation

and presence in the field. The MLF has been accused of being a statistically unclear black box algorithm because of the nature of the unexplainable way it reaches a decision. The other two algorithms are empirical and are not based on explicit theories. The MLF and the California algorithms use from one to several persistence intervals to suppress the effects of minor traffic fluctuations. The Minnesota algorithm achieves the same goal by smoothing and filtering techniques. None of the algorithms is designed to handle freeway bottlenecks due to an upstream demand that exceeds the existing capacity of the given section. Virtual bottlenecks due to consistent loop biases are not accounted for either. Table 2.1 summarizes the previous evaluation of the universality of the three algorithms, or more precisely the lack of.

## **2.5 THE NEED FOR A UNIVERSAL FRAMEWORK**

It is evident from the previous section that none of the existing algorithms is universal. Further improvements are needed for an AID algorithm to be deployable, transferable, and successful. It is therefore the focus of this research to produce such an algorithm. The development of the envisioned algorithm will be guided by the universality requirements defined above. The universality requirements list is by no means collectively exhaustive. It only captures the most important and pressing issues pertinent to the problem of automated incident detection on freeways.

Universality Attribute	Algorithm		
	MLF	CA #8	Minnesota-sota
1. Highest performance.	√	x	x
2. Fast, robust and automated training and retraining	x	x	x
3. Reasonable TMC implementation requirements	x	x	x
4. Transferable logic:	√	√	√
5. Reasonably transferable training / calibration parameters	x	x	x
6. Minimal initial training data requirements	x	x	x
7. Account for prior probabilities of incidents	x	x	x
8. Account for the unequal costs of misclassifying traffic patterns	x	x	x
9. Capable of producing the posterior probability of an incident	%	x	x
10. Estimate incident severity	%	x	x
11. Capture incident duration	%	%	%
12. Statistical and theoretical soundness and clarity	x	N.A.	N.A.
13. Immunity to minor traffic fluctuations effects	√	√	√
14. Immunity to bottleneck effects	x	x	x
15. Immunity to consistent loop detector biases	x	x	x

x: absent attribute

√: fulfilled attribute

?: attribute fulfilled to some extent, but not fully.

N.A.: Not Applicable.

**Table 2.1. Lack of Universality of the MLF, CA, and Minnesota Algorithms**

## **CHAPTER 3**

### **FREEWAY STUDY SITES, INCIDENT AND LOOP DETECTOR DATABASES**

#### **3.1 INTRODUCTION**

The freeway sites chosen for this study and the associated incident and loop detector databases are described in this chapter. First, the freeway test sites, their geometry and instrumentation are described. The nature and details of the available incident databases that represent the ground truth for the study as well as the process of extraction of the corresponding incident-related loop detector data are then discussed.

#### **3.2 TEST SITE SELECTION**

Three primary study sites and sources of incident and loop detector data have been focused upon in this research. The three sites are: the I-880 (Nimitz) freeway in Alameda County, California, the I-35W freeway in Minneapolis, Minnesota, and the SR-91 freeway in Orange County, California. These sites have been selected due to their diverse geometric configurations and the availability of traffic and incident information.

### **3.2.1 The I-880 Freeway Test Site**

Data from the first site has been recently made available in California. This database is the result of a comprehensive data collection effort on a fully instrumented section of the I-880 (Nimitz Freeway) in Alameda County, California. The length of the selected section of the I-880 freeway is 49700 feet between the Marina exit and Wipple exit as shown in Figure 3.1. The freeway section is instrumented with inductive loops buried in the pavement. The instrumentation, however, is confined between the Lewelling and Industrial exits. There are a total of 18 loop stations covering all lanes of the freeway and selected on and off ramps. The spacing between the different loop stations ranges from approximately 1000 feet to 3300 feet. On the main line lanes the detectors are placed in pairs, but on the on and off ramps they are single detectors. The data collected were vehicle counts, occupancy, speeds, and loop-on times. Software specially developed at the University of California, Berkeley can be used for converting the binary loop data into readable ASCII format and for fixing any abnormalities.

### **3.2.3 The I-35W Freeway Test Site**

The second study site is the Interstate 35W Freeway in Minneapolis, Minnesota. The site is 12.5 miles in length between Co. Rd. 36 and the I-494

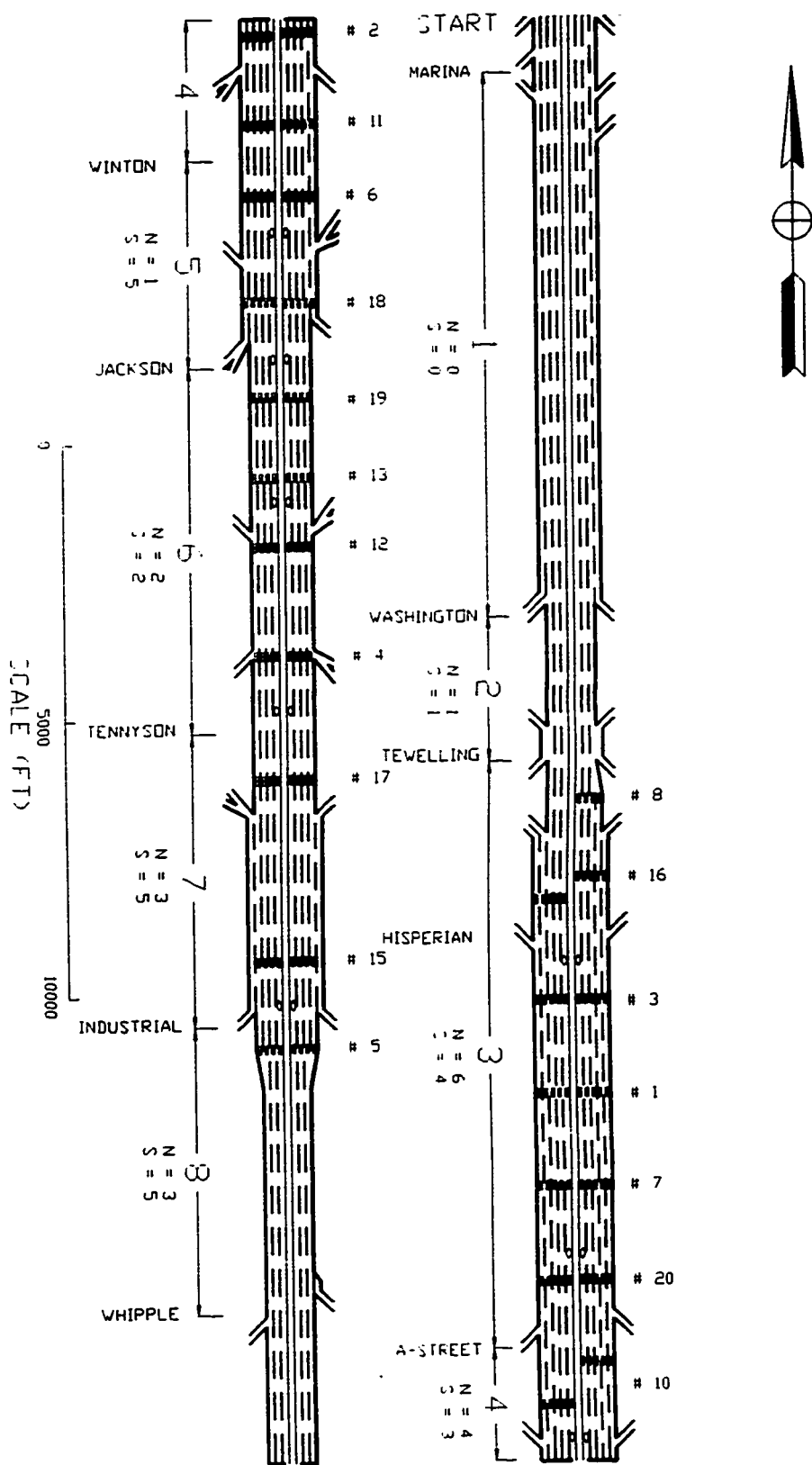


Figure 3.1. The I-880 Freeway Site

freeway. This freeway section is heavily traveled and is often congested. The section is instrumented with both loops and cameras. It has three lanes along most of its length. It includes 4 major bottlenecks. Loops are 0.3-0.7 miles apart. The locations of detector stations and cameras are shown in Figure 3.2.

### **3.2.3 The SR-91 Freeway Test Site**

The section selected from the SR-91 Riverside Freeway in Orange County, California, is the westbound direction between the SR-57 and Interstate 5 Freeways. The average daily traffic on this study site is approximately 200,000 vehicles per day, both directions. The entire site is over 5.0 miles in length and has eight loop detector stations. The spacing of detector stations varies from 0.34 to 1.02 miles. A schematic showing the lane geometry, together with detector station locations is shown in Figure 3.3.

As described above, all three sites have a variety of detector spacings, distances between on-ramps and off-ramps, lane geometries, lane detector instrumentation and bottleneck configurations, making them good locations to test the performance of AID algorithms under a variety of conditions. All the loop detector data used in this study from the sites above were 30 seconds averages across all lanes of volumes and occupancies. Using the detector

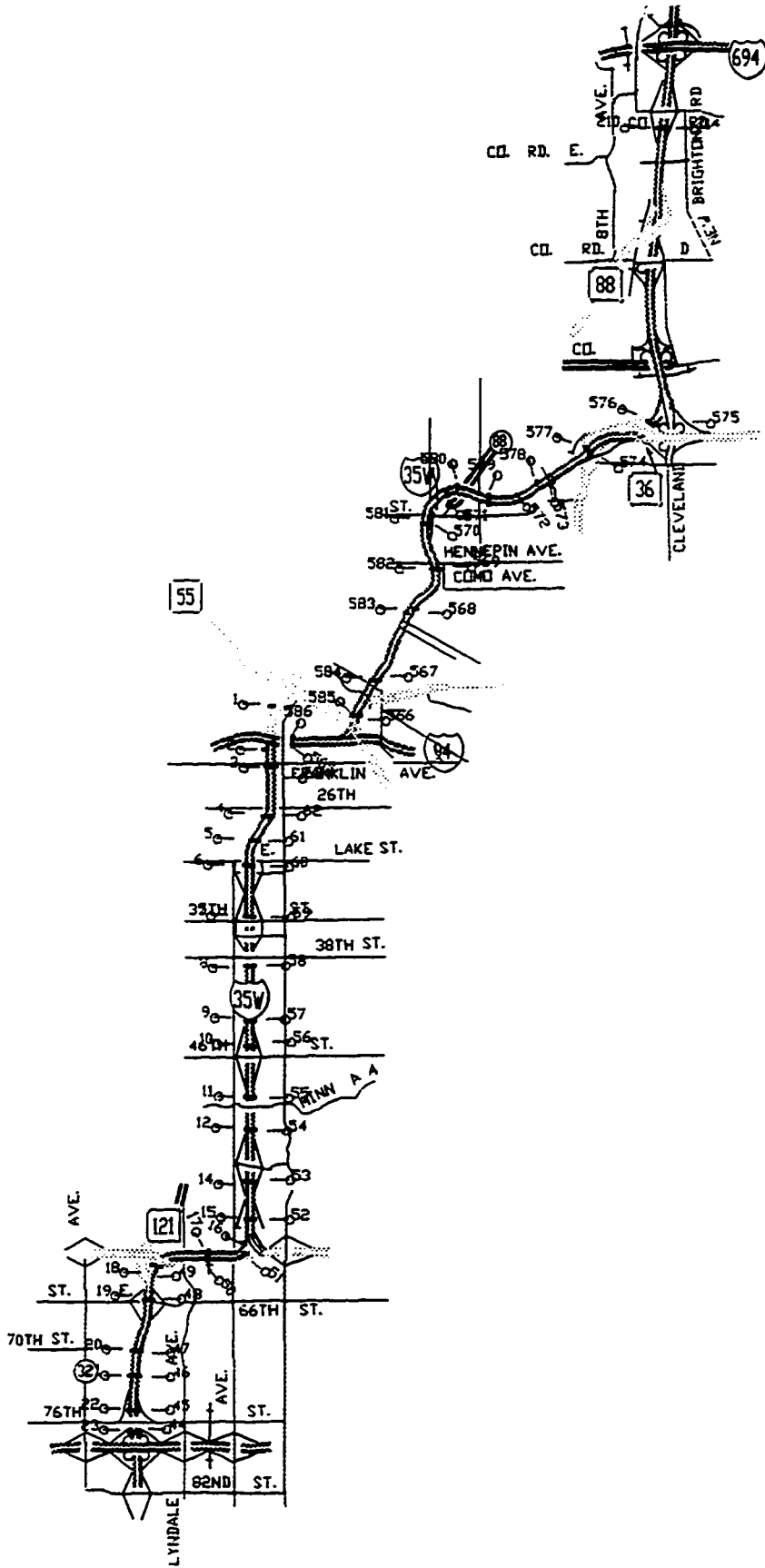


Figure 3.2. The I-35W Freeway Site



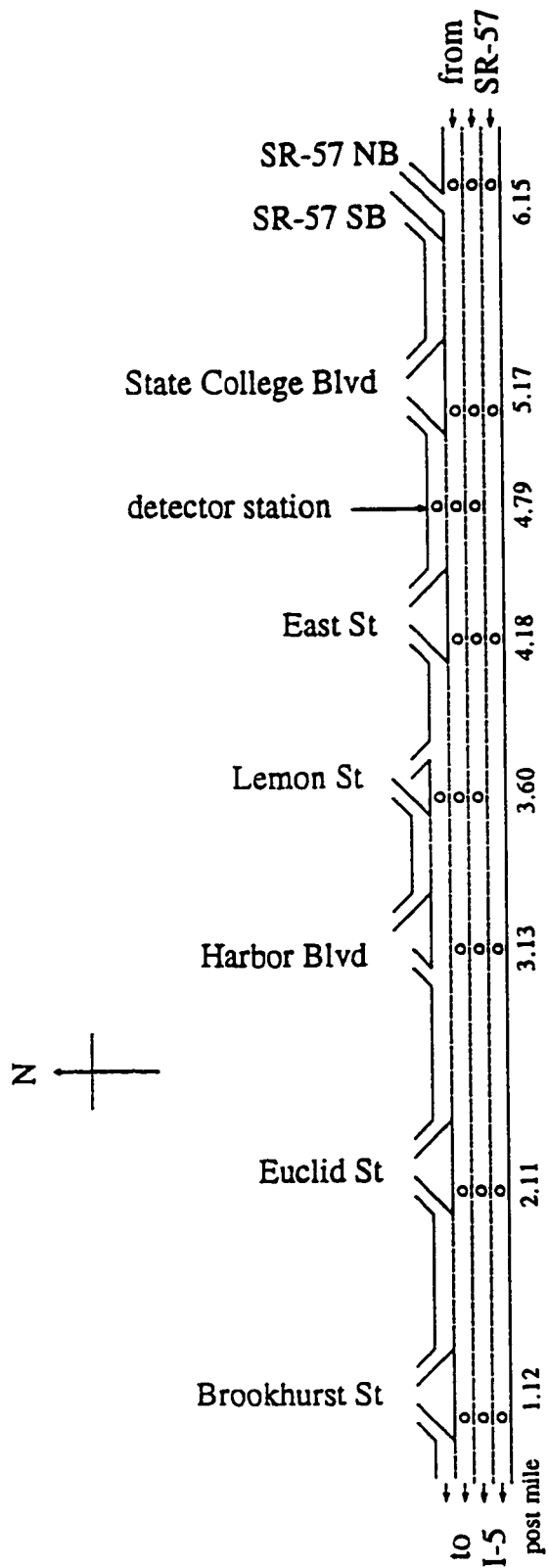


Figure 3.3. The SR-91 Freeway Site

stations as boundaries, all study sites were divided into segments or sections. Each segment is bounded by detector stations at the upstream and downstream ends, referred to as the upstream and downstream stations, respectively. Detector data from both stations were utilized to form the input features to all the AID algorithms under consideration. Incidents occurring at the fringes of the test sites for which only data from one loop station were available were excluded from the study.

### **3.3 TRAFFIC EVENT DATABASES DEVELOPMENT**

#### **3.3.1 The I-880 Database**

The loop data consists of the output from different loop detectors arranged on the freeway section. There is one binary (non-ASCII) file per cabinet per day which is not readable. Therefore, the FSP software (described below) was developed at U.C. Berkeley in order to analyze the data. Pairs of loops, fourteen feet apart, known as speed traps, were available on the main lanes only. On the on and off ramps there were single detectors only. From the data the program calculates the number of cars that pass over the detectors, their average speed, and the average occupancy per period. There were times when the loops failed. For one reason or another, a loop would go out periodically or count incorrectly. Thus, two fixes were incorporated in the

FSP program developed at Berkeley in an attempt to restore the integrity of the data. The first fix is termed a hole fix, which tries to fill in any missing data by recreating data based on upstream and/or downstream data. The second fix, the consistency fix, attempts to correct systematic errors in the data, such as over or under counting. Using these fixes, more accurate loop data was obtained.

The incident database was collected via drivers of the probe vehicles which combed the freeway section during the entire data collection period. When a driver passed an incident, he/she would radio the information to a command center for all entries to be logged. The log data was coded into a standard form for input into the database. Problems encountered with the incident database included the accuracy of the location of the incidents and the starting and ending times. To fix the first anomaly, the incident database was correlated with the probe vehicle database to try and pinpoint the location of the incident. In the probe vehicle database, as drivers traversed the highway they pressed keys on portable computers indicating their position. The starting and ending times were fixed in a similar manner. By using the loop data in conjunction with the incident and car data, a comprehensive data set emerged.

Analysis of the I-880 incident data revealed that only 45 lane-blocking incidents produced congestion shockwaves and hence only 45 such incidents

were usable. Some lane blocking incidents occurred at the fringe of the instrumented section and hence did not have both an upstream and downstream loop station. Such incidents were not used in this study. The lane-blocking incidents are good candidates for training and testing the incident detection algorithms because they usually produce congestion and their effect on the traffic is detectable by loops. No attempt was made to locate non-lane blocking but detectable incidents because of the impracticality of doing so in such a large database with thousands of non-lane-blocking incidents. Among the 45 lane blocking incidents that were actually used, 15 incidents involved two or multi-vehicle collision resulting in a lane blockage for a significant amount of time. Twenty five, mostly multi-lane blocking incidents occurred in the south bound direction while 20 lane blocking incidents occurred in the northbound direction mostly with one blocked lane.

After the necessary fixes were made using the FSP program, a second phase of visual analysis was conducted in order to accurately identify the starting and ending times of each incident as well as the exact location. Three dimensional graphs of occupancy-time-space were generated for every day in which an incident was reported. The well known "Matlab" software was used for this purpose. After the exact time and location of each incident were identified, the corresponding loop data were extracted into separate files and reformatted to suit the requirements of the different incident detection

algorithms under investigation. It should be noted again that the extracted data were average values across all lanes.

### **3.3.2 The I-35W Database**

Data from this test site were collected over a period of 4 months, 7 days a week, and 24 hours a day. The data collection process was administered by the University of Minnesota. During the data collection period, 159 incidents, 94 south bound and 65 northbound were reported by the traffic operator and entered in the incident database that represented the ground truth of the study. The Minnesota database includes usual information about location, date, time, and incident duration. It also includes information on the type of incident and distinguishes amongst 10 types: accidents, stall, lane closure, flat tire, mechanical, vehicle fire, patrol, roll over, spilled load, and spinout.

The number and identity of the lanes affected by each incident are recorded, and the roadway conditions reported as dry, wet, snow-ice, or including construction. The database also includes a five-level entry describing the impact of the incident on the traffic. The levels are: no impact, affecting traffic but no congestion, resulted in limited congestion, resulted in severe congestion, and added to existing problem. Additional information in the data base includes the time at which all the vehicles were pushed to the

shoulder and the “all clear” time marking the departure of all vehicles involved.

Confirmation of incidents was accomplished mostly through CCTV cameras along the freeway segment. Of the 159 incidents, 135 were classified according to the impact they had on traffic. More specifically, 65 were found to have an impact that was an addition to the existing problem, 6 resulted in severe congestion, 27 limited congestion, 29 affected the traffic but without congestion, and 8 had no impact on traffic. Within the 159 incidents there were 68 accidents, 44 southbound and 24 northbound. Of the 68 accidents, 2 resulted in severe congestion, 16 limited congestion, 13 affected the traffic but without congestion, 3 had no impact on the traffic, 31 were an addition to an existing problem, and 5 were not classified. The incident database also included 76 “stall” vehicles. Almost all stalls occurred on the shoulder, 8 resulted in limited congestion, 14 affected the traffic but without congestion, 5 had no impact on the traffic, 38 were an addition to an existing problem, and 11 were unclassified.

As was the case with the I-880 data, the reported location and time of each incident were found to be not accurate. Some loop data were also missing as reflected by “-9” entries in the data files. No automated process existed for fixing holes in the loop data and inaccuracies in the incident data. The entire data set had to be manually reviewed, incident by incident, in order to

pinpoint the exact starting and ending time of each incident, the exact location and the immediate upstream and downstream loop stations. Data from malfunctioning loops were taken out from the database. Finally, incidents that occurred at almost the same time and location, and hence resulted one congestion shockwave, were combined as one event.

### **3.3.3 The SR-91 Database**

Data from the SR-91 freeway can be divided into two parts: the large simulation data set was prepared by Cheu (1994), and a small set of real data obtained from Caltrans and the California Highway Patrol. The following is only a brief summary.

The simulated data set was based on loop detector data generated from INTRAS, a microscopic freeway traffic simulation model (Wicks and Lieberman, 1980). INTRAS was calibrated with field data collected from the study site during incident-free and incident conditions (Cheu et al., 1993). A total of 400 simulations were made, each with a lane-blocking incident, under a range of peak and off-peak volumes. These incident scenarios were constructed as follows. First, 50 incidents were assigned to each of the seven freeway sections in the study site, with the addition of 25 incidents each upstream and downstream of the study site. Within a freeway section, the longitudinal positions of the incidents were randomly distributed. For each

position, the pattern of lane blockage was randomly selected from a population of possible blockage patterns based on the lane geometry at that location. For each freeway section, the volume and occupancy accumulated over 30-second intervals at the upstream and downstream stations were extracted to form the data set. The detector output of the seven sections was combined to form the training data set 1, which had 35000 input vectors (of 30-second data). Independent data sets (data sets 2 & 3) were also generated with different random number seeds and were used to monitor the progress of training, and test the trained networks, respectively.

The real data set consisted of only nine incidents. In addition, a total of 63 hours of real incident-free data was derived to test for false alarms. Of the nine incidents in this data set, one occurred on the shoulder, five resulted in one-lane blockages, two involved two-lane blockages, and one had no reported details on the extent of lane blockage.

### **3.4 CONCLUDING REMARK**

One of the main objectives of this study is to heavily utilize several recent unprecedented large real incident and related loop databases. These were obtained from the I-880 freeway and the I-35W freeway. The large simulated database and the limited real data base from the SR-91 freeway were also included in the study for three main reasons. First, they were available.



Second, they provide the opportunity to test the transferability of all algorithms from a simulated environment to a real one. Finally, they facilitate cross-comparison of the results between this study and the previous ANN-related study by Cheu (1994). Such a large variety of data is very useful for the results to be comprehensive and comparable and neither site-specific nor study-specific.

## **CHAPTER 4**

### **DETECTABLE INCIDENTS AND ACCEPTABLE LIMITS ON DETECTION PERFORMANCE**

#### **4.1 INTRODUCTION**

The most widely known and used evaluation criteria for AID algorithms are the Detection Rate, False Alarm Rate, and mean Time to Detection. These are also utilized in this study. However, no prior published attempt has been made to establish operational boundaries on these criteria that would define the border line between acceptable and unacceptable performance of a particular AID algorithm. This problem greatly hinders the interpretation of any AID results on an absolute scale, and limits their usefulness to comparative evaluations only. Consequently, and regardless of the specific performance results obtained from evaluating any AID algorithm, it remains largely unknown whether this performance would be acceptable to practitioners in the field or not. Therefore, this chapter is dedicated to such needed a priori definitions. The basic definition of what is an incident, from a traffic management perspective is first presented. The performance measures used for evaluating the detection performance of the different AID algorithms are then explicitly defined. Finally, a selective TMC-survey to elicit an acceptable set of limits on these performance measures is described.

## 4.2 WHAT IS AN INCIDENT?

Despite the long history of incident detection research, the definition and classification of incidents remain debatable. Depending on the perspective, different definitions arise. From a traffic safety perspective, an incident is any non-recurring event that could pose a hazard to motorists. This category includes events such as accidents, disabled vehicles, spilled loads, and temporary maintenance and construction activities. From a traffic management perspective, the definition would be similar but limited to the cases causing unexpected congestion shockwaves, queues and delays. If maintenance activities are planned ahead of time, then they are not unexpected and hence should be excluded from the definition. If an incident causes no congestion, then it should be excluded as well. Since AID is primarily a traffic management tool, it seems plausible therefore to define an incident as “any unexpected non-recurring event that causes disruption to the normal behavior of traffic, producing congestion shockwaves, queues, and motorist delays”.

## 4.3. DETECTABLE CLASSES OF INCIDENTS

In general freeway traffic disturbances could be grouped into two major categories (Ritchie and Stephanedes, 1996): incidents on the freeway, and all

other disturbances that are not directly associated with an incident, including bottlenecks, traffic pulses in uncongested flow, compression waves in congested flow and random traffic fluctuations. The disturbances in the second category are the ones that usually present problems to traffic engineers in the development and implementation of incident detection algorithms. This is attributable to the nature of the flow patterns they create which can be similar to incident patterns. Existing algorithms confuse these patterns, especially bottlenecks, with incidents and generate false alarms.

As discussed earlier, incidents unexpectedly reduce roadway capacity at the affected location. If the reduced capacity is less than the upstream demand, two regimes of traffic conditions are created, congested flow upstream (high occupancies) and uncongested downstream (low occupancies). Two shock waves are generated and propagate upstream and downstream, each accompanying its respective regime. The boundary of the congested region propagates upstream at an approximate speed of 10 miles per hour, where the exact value depends on the characteristics of the incident, the geometry of the freeway segment and the level of traffic at the time of the incident. Downstream of the incident, the freeway is cleared of traffic; the boundary of the cleared region propagates downstream at a higher speed.

Capacity reducing incidents are usually lane-blocking ones. This fact leads to the usual classification of incidents into lane-blocking and non-lane

blocking ones. However, real data reveals that some shoulder incidents and incidents on the opposite side of the freeway that involve police and emergency vehicles could also cause capacity reductions due to rubbernecking and the fact that drivers behave extra-cautiously while passing through the affected section. Thus, for an incident to be detectable using a loop-detector-based system, it does not have to be restricted to the lane blocking class, but must cause a capacity reduction. Moreover, the reduced capacity must be less than the upstream demand, causing a temporal bottle neck and propagating congestion shockwaves to the upstream. On the other hand, however, lane blocking incidents are easier to identify in a large incident database by knowing the lane at which the incident occurred. It is very difficult for researchers to objectively identify all “sensible” or detectable incidents, and hence they may reasonably restrict themselves to lane-blocking ones. An incident detection system that is trained to capture lane blocking incidents is in fact capable of capturing the more general class of detectable incidents. Lane blocking incidents are used as a mere representative of the larger class of detectable incidents. Nonetheless, adding non-lane-blocking incidents to the training process of AID algorithms would certainly help improve their performance. It is worthy to mention at this point that of the databases used in this study, the I-880 database comprises strictly lane-blocking incidents, while on the other hand,

the Minnesota database contains both lane-blocking and non-lane-blocking incidents.

Bottlenecks are formed where the freeway demand/capacity ratio changes, e.g., reduced capacity due to lane drops, or a significant demand change at an entrance ramp with a substantial traffic input, or at an off-ramp with a substantial traffic discharge. Like incidents, bottlenecks create a discontinuity in the traffic parameters very similar to that created by incidents, only lasting longer. Such bottlenecks are a primary cause of false alarms and they need special attention while designing AID algorithms. This problem will be addressed in due course in this research. Traffic pulses are observed in uncongested flow and are created by platoons of cars moving downstream. Compression waves occur in heavy, congested traffic, usually following a small disturbance, and are associated with severe "slow-down and speed-up" vehicle speed cycles. Waves are typically manifested by a sudden, large increase in occupancy that propagates through the traffic stream in a direction counter to the traffic flow as time progresses. Data reveal that compression waves result in station occupancies of the same large magnitude as in incident patterns. Compression waves constitute a primary source of false incident alarms. Random fluctuations are often observed in the traffic stream as short-duration peaks of traffic occupancy. These fluctuations, although usually not large in magnitude, may occasionally form an incident pattern or obscure real incident patterns.

Detection system failures could also cause false alarms. For instance, a stuck upstream sensor in the “on” position or a downstream sensor in the “off” position could cause spatial patterns similar to incidents. Consistently biased sensor reading (higher or lower than actual), could cause the same problem.

To summarize, this research focuses on developing AID algorithms that are capable of detecting shockwave-producing incidents, both lane-blocking and non-lane-blocking, and without confusing incident-similar patterns caused by bottlenecks and loop biases for an incident. Achieving this objective would improve incident detection in two ways, first by generalizing the detection capability to include detectable non-lane blocking incidents, and second by alleviating the confusion caused by physical and virtual bottleneck situations.

#### 4.4 PERFORMANCE MEASURES

The performance measures used are the Detection Rate (DR), False Alarm Rate (FAR) and Time to Detection (TTD). The DR is defined as

$$DR = \frac{\text{no. of detected incidents}}{\text{total no. of incidents in the data set}} \times 100\%$$

If the algorithm issues an incident warning for a freeway location at a particular time, but in absence of an actual incident in the field, a false alarm is said to have occurred. The FAR is generally defined as

$$\text{FAR} = \frac{\text{no. of false alarms}}{\text{total no. of applications of the algorithms}} \times 100\%$$

The TTD of an incident is the time between the actual occurrence of the incident and the time it is detected by the algorithm. For multiple incidents in the data set, the mean TTD of the detected incidents is always used. For further details the reader is referred to (Cheu, 1994).

#### **4.5 TMC-ACCEPTABLE LIMITS ON DETECTION PERFORMANCE**

Despite research attempts to develop the 'perfect' incident detection algorithm that would yield 100% Detection Rate (DR) and 0% False Alarm Rate (FAR), with acceptable TTD, this ultimate performance has not been reached yet. Even if such performance could be achieved on one particular database, it would be improper to draw generalized conclusions. Therefore, there always exists a trade-off between a low FAR and a high DR. It is not clear which is more important to achieve at the expense of the other, high DR or low FAR. Different Traffic Management Centers (TMCs) are expected to have different emphases, requirements and constraints on the acceptable limits of DR and FAR. Identifying or drawing representative limits on the



performance measures is therefore concluded to be of high priority in order to assist in evaluating AID algorithm performance. Towards this goal, the following attempt was made to elicit such operational constraints and criteria from freeway operations personnel. A short survey was prepared and sent out to 15 TMCs all over the US that were thought to be interested in automated freeway incident detection. The actual survey form is shown in Figure 4.1. The three questions in the survey targeted the identification of what might be an acceptable maximum limit on FAR and minimum limit on DR. First the purpose of the survey was explained as well as the definition of the terminology used together with representative numerical examples. The first question aimed at sensing the general preference of the subject TMC regarding high DR or low FAR. The second question presented to the TMC personnel the actual performance of a variety of existing incident detection algorithms, to see if any of the algorithms would be considered usable, and again to detect their inclination towards either high DR or low FAR. The third and last question asked for explicit specification of the absolute limits on the evaluation variables in terms of an upper limit on the FAR, above which the algorithm would be an unbearable 'crying wolf' and a lower limit on the DR below which it would not be performing its intended function of detecting incidents.

Seven replies were received to the date of this writing. Analysis of the different replies indicated that, on the average, TMC personnel put more

emphasis on a high DR, which is believed to be consistent with the prime objective of incident detection. Out of the seven responses received, four TMCs (57%) stated their preference for a high DR in response to question #1 in the survey form. In response to question #2, four TMCs chose system #1 which has a 100% DR despite of the relatively high FAR of 5%. Only one TMC picked system #2, and the same for systems #3, and #4. The response to the third question, which involved an explicit statement of the acceptable boundaries on DR and FAR, varied from a TMC to another with an average requirement of the DR to be at least 88.3% and of the FAR to be at most 1.8%. The actual responses are listed in table 4.1.

								Average	Extreme
DR% at least:	80	95	max. <sup>a</sup>	70	90	95	100	88.3%	100
FAR% at most:	1	5	0.25	2	0.5	2	min. <sup>b</sup>	1.8%	0.25

<sup>a</sup> max attainable.

<sup>b</sup> min attainable.

**Table 4.1. DR and FAR Limits Extracted from TMCs' Responses.**

The above results indicate that a reasonable set of limits on DR and FAR would be 88% and 1.8% respectively. A more stringent set of limits could be obtained from using the extreme value limits of 100% and 0.25% respectively as shown in Table 4.1. All algorithms under consideration in this research will be evaluated against both the two sets of limits.

UNIVERSITY OF CALIFORNIA, IRVINE

BERKELEY • DAVIS • IRVINE • LOS ANGELES • RIVERSIDE • SAN DIEGO • SAN FRANCISCO



SANTA BARBARA • SANTA CRUZ

INSTITUTE OF TRANSPORTATION STUDIES

IRVINE, CALIFORNIA 92717-3600  
 (714) 824-5989  
 FAX (714) 824-8385

Despite research attempts to develop the 'perfect' incident detection algorithm that would yield 100% Detection Rate (DR) and 0% False Alarm Rate (FAR), this ultimate performance has not been reached yet. There exists a trade-off between a low FAR and a high DR. We define DR as the ratio of the number of incidents successfully detected to the total number of incidents that have occurred. Also the FAR is defined as the ratio of the number of times the algorithm falsely indicates an incident to the total number of times the algorithm is applied (usually once every 30 sec.). For instance, if 10 incidents occur, and the algorithm detects only 8 of them, then the DR is 80%. The FAR computation is a little bit more tricky. For instance, in one hour the algorithm is applied 120 times at a given section. If the algorithm indicates an incident 3 times during this hour while there are no incidents in the field, the FAR is  $3/120 = 2.5\%$ .

Please answer the following three questions related to DR and FAR:

- Which is more important to you? please do not answer 'equally important'.  
 Low FAR      \_\_\_  
 High DR      \_\_\_
- If you have access to different incident detection algorithms with the following performances, which algorithm would you be most likely to implement?  
 #1 \_\_\_      #2 \_\_\_      #3 \_\_\_      none of them \_\_\_

	Algo #1	Algo #2	Algo #3
DR%	100%	70%	50%
FAR %	5%	1%	0.25%
Average number of false alarms per hour per mile*	12	2.4	0.6

\* based on loop stations .5 mile apart and 30 sec apart applications

- Please give percentages for the boundaries of DR and FAR that you believe are acceptable. (For instance DR should be at least 90% and FAR should be at most 1%)  
 DR should be at least:      \_\_\_%  
 FAR should be at most:      \_\_\_%

Traffic Management Center: _____	
Contact Person Name & Title: _____	
Signature: _____	Date: _____

Figure 4.1. Survey Form Mailed to Different TMCs.

## **CHAPTER 5**

### **INVESTIGATING ADVANCED NEURAL NETWORKS FOR IMPROVED INCIDENT DETECTION**

#### **5.1. INTRODUCTION**

This chapter is not meant to be an introduction to the basics of neural networks, nor to their use for incident detection. A more introductory treatment of both subjects can be found in Cheu (1994). The focus instead is on advanced neural network architectures that have the potential to produce the improved incident detection framework discussed earlier. Of the universality requirements, the speed, and robustness of the training process, and the potential of automating such a process are of prime importance. Such qualities would allow for on-line adaptation of the resulting AID algorithm to the local freeway site conditions. Hence, an algorithm trained to be generic at first, could automatically tailor its parameters to the site-specific needs without requiring attended retraining. Two promising new models are investigated in this chapter, namely, the LOGICON™ Projection Neural Network, and the Probabilistic Neural Network (PNN). The theory underlying each network is presented first. Both networks are then trained and tested on simulation data in order to facilitate comparison to the former results from the Multi-layer Feed Forward (MLF) neural network used by Cheu (1994). Finally, the more promising of the two architectures, in

terms of universality, will be chosen for detailed investigation in the rest of study using primarily real data.

## 5.2 THE LOGICON PROJECTION NETWORK™

### 5.2.1 Theory

The Logicon Projection Network™ was first introduced by Gregg, and Narbik (1992) and is the subject of a patent application by Logicon RDA. It is incorporated in the NeuralWare neural networks software package (1993) for non-commercial research only. Details of the network can be found in the above references. The following is a brief overview.

The prime motivation behind investigating this architecture is its faster training as opposed to the MLF. The advanced input preprocessing capabilities of this network architecture are inspired from a stereographic projection of input patterns (James and Chris, 1991). The input pattern set is transformed into another set where all the patterns lie on the surface of a sphere (or any other body with positive curvature). This inevitably involves increasing the dimensionality of the input space by unity, i.e. an  $N$ -dimensional pattern vector  $X$  is transformed into an  $(N+1)$ -dimensional vector  $X'$  subject to the constraint  $|X'| = R$ , where  $R$  is the radius of the hypersphere.

The projected or transformed vectors serve as inputs to a conventional MLF with  $(N+1)$  nodes in the input layer instead of just  $N$  nodes as shown in Figure 5.1. The geometrical interpretation of such a projection is shown in Figure 5.2 for the case of 2-D inputs projected onto a 3-D sphere. The vector  $X$  connects the origin of the 2-D space to the point  $X=(x_1, x_2)$ . The 3-D vector  $X'$  is obtained by connecting the center of the sphere and the point  $X$  and extending the line to intersect the surface of the sphere.

As  $X'$  is 3-D ( or  $(N+1)$ -dimensional ), the weight vector  $W'$  that connects the modified input to the hidden layer is also 3-D. The net input to a hidden layer node is :

$$I = W'.X' - S$$

where  $S$  is the threshold for that node.

The output of the node is a non-linear function of the net input  $O(I)$ . A sigmoidal or hypertangent function is often used. Obviously, the hidden nodal output has a constant value when the nodal input has a constant value. For instance, an intermediate node has an activation (output) value of  $1/2$  when the input to that node is  $0$ . Thus, all inputs with  $0$  value define a surface of a constant output of  $1/2$ . Such a decision surface divides the input space into two: input vectors which lie on one side of this surface will produce activation greater than  $1/2$  and vice versa. In the original  $N$ -

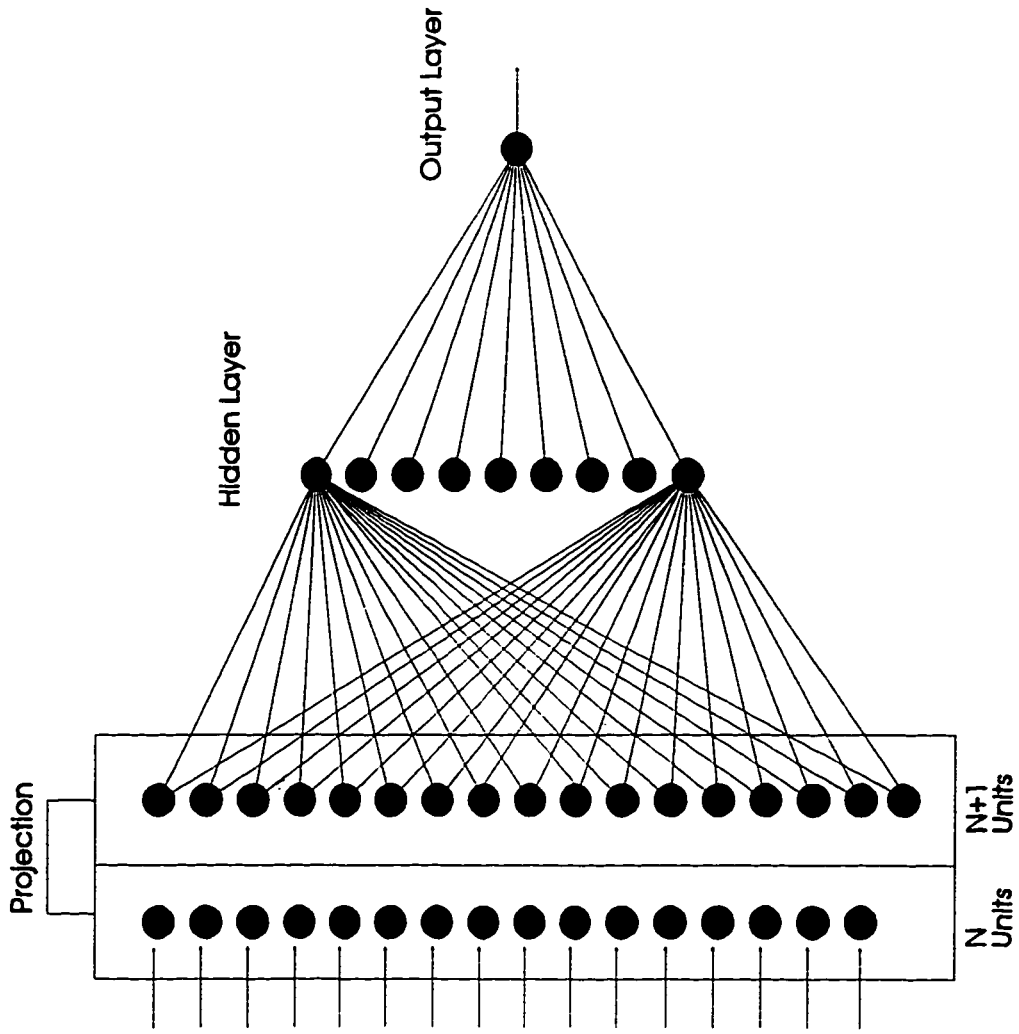
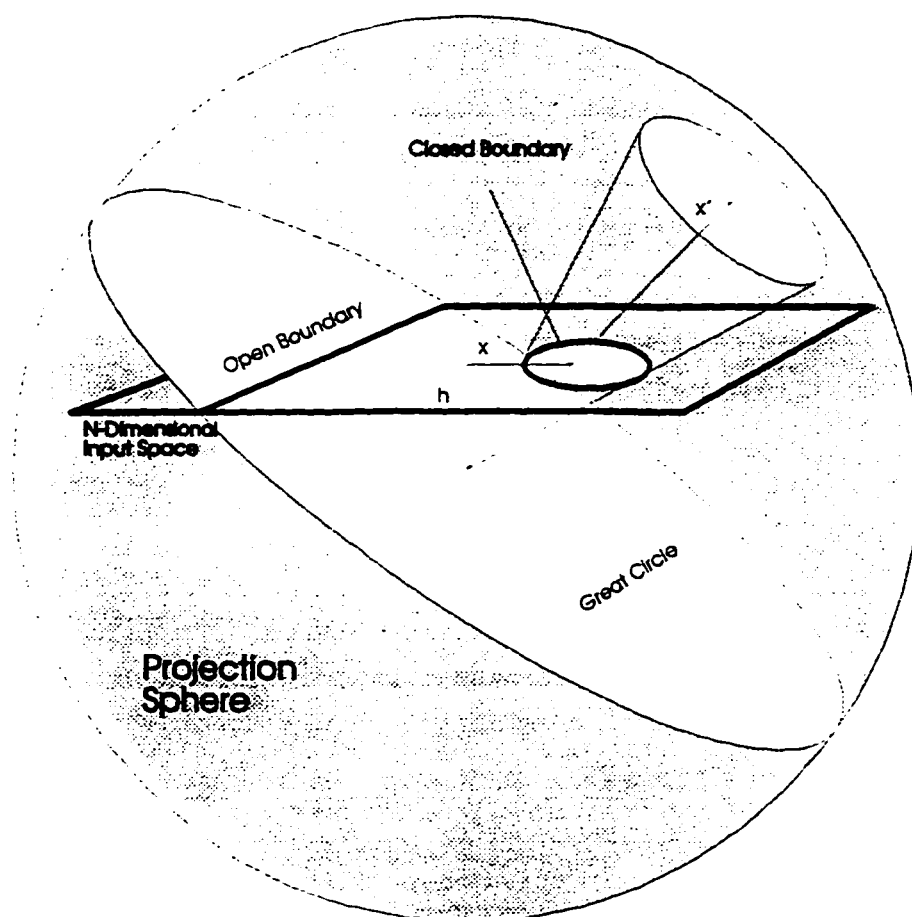


Figure 5.1. The Logicon Projection Network™ (Gregg and Narbik, 1992)



**Figure 5.2. Stereographic projection in the Logicon network (Gregg and Narbik, 1992)**



dimensional space, this decision surface is a hyperplane (a line in 2-D) described by :

$$W.X - S = \text{constant.}$$

The constant is 0 when the decision surface is chosen to correspond to an activation of 1/2. In general, for a given activation and hence a given value of the constant, the threshold  $S$  determines the location of the hyperplane. With the constant chosen to be 0, the threshold is proportional to the distance of the hyper plane from the origin.

In  $(N+1)$  dimensions, each hidden layer node still draws a hyperplanner decision boundary that intersects with the hypersphere around  $X'$ . This is a circle around  $X'$  in Figure 5.2. It should be emphasized that the threshold  $S$  determines the position of this intersection, consequently the projection of the surface resulting from the intersection back onto the original  $N$ -dimensional space is a function of the threshold  $S$ . In Figure 5.2, for instance, if  $S$  is large, the resulting intersection circle is small and lies on one side of the original 2-D plane. The projection of that circle on the 2-D plane is an ellipse (closed boundary). On the other hand, if  $S$  is small, the intersection circle approaches a great circle on the sphere and its projection back on the 2-D plane is a curve or ultimately a line (open boundary).

It is this ability to form hyperplanner or hyperspherical prototypes that allows the Logicon network to be initialized rapidly to a good starting point which is already close to a desirable minimum. Furthermore, during learning, closed boundaries can become open ones and vice versa as the weights and thresholds are adjusted.

### **5.2.2 Advantages of the Logicon Network**

The MLF network suffers from slow training times, the potential to get stuck at local error minima, and the need for a large number of nodes when applied to complicated problems. However, in problems for which it does converge to a solution, it offers the advantage of ensuring error minimization. On the other hand, networks that start by placing a prototype train quickly but do not guarantee minimization of the classification error (Gregg and Narbik., 1992). The Logicon network has the following advantages (NeuralWare 1993):

- It combines speed of hyperspherical networks with error minimization of the MLF
- It learns faster (two orders of magnitude) by properly initializing the network weights and thresholds to prototypes of the input set.

- It partitions the input space into localized regions which allows for easy separation of the inputs into different classes.
- It uses hidden layer nodes more efficiently (saves them for more complex problems).
- It is quasi modular, allowing the combination of two or more networks or the addition of new input classes and ranges.
- The advantage of speed-up and efficiency are enhanced for more complex problems and higher dimensional inputs.

For the purpose of incident detection in particular, the Logicon network's fast training attribute is promising. As mentioned earlier, this could allow for on-line adaptation of the resulting AID algorithm to the local conditions. The network could be trained to be generic at first and then the fast training capability could be capitalized upon in order to automatically tailor its parameters to the site-specific needs.

### **5.3 THE PROBABILISTIC NEURAL NETWORK (PNN)**

The Probabilistic Neural Network (PNN) is a pattern classifier that combines the well known Bayes strategy for decision making with the also well known Parzen non-parametric estimator of the probability density functions of the

different classes. The PNN-variants used in this study are based on the original PNN architectures introduced and successively enhanced by Specht (1990, 1992, 1994, 1996). Unlike the MLF-family of networks, the PNN-family asymptotically approaches the Bayes optimal decision surface without the danger of getting trapped in local minima. It is also orders of magnitude faster to train than the MLF-family of networks.

### 5.3.1 Theory: Multivariate Bayesian Discrimination and the PNN

The objectives of the PNN as well as any other pattern classifier are to: [1] separate classes of objects, i.e. define the boundaries between the existing classes, and [2] classify new objects to one of the existing classes in a way that minimizes the expected risk. An object is defined by a vector in a  $p$ -dimensional input space, where  $p$  is the number of features or variables. In the following sections the mathematics will be explained for the case of 2-classes in 2-dimensions. Extension to higher cases can be done in a straight forward manner without loss of generality.

Let  $f_1(x)$  and  $f_2(x)$  be the probability density functions (PDFs) associated with the  $p$ -dimensional input vector  $X$  for the populations  $\pi_1$  and  $\pi_2$  respectively. The Bayes classification rule that minimizes the Expected Cost of Misclassification (ECM) is to assign a new vector to either class  $\pi_1$  or class  $\pi_2$

based on the density ratio, the misclassification cost ratio, and the prior probability ratio as follows :

X belongs to:

$$\pi_1 \text{ if : } f_1(x) / f_2(x) \geq \{ [ C(1|2) / C(2|1) ] * [ P_2 / P_1 ] \}$$

$\pi_2$  otherwise.

where :

- $C(i|j)$  is the cost of misclassifying an object as belonging to population  $\pi_i$  while it belongs to population  $\pi_j$ .
- $P_i$  is the prior probability of occurrence of population  $\pi_i$

The key to using the above classification is the ability to estimate the PDFs based on training patterns. Often the a priori probability can be estimated, and the cost ratio requires subjective evaluation.

The accuracy of the decision boundaries and the subsequent classification depends on the accuracy with which the underlying PDFs are estimated. A good feature of this approach and the related PNN implementation is the PDF estimation consistency. Consistency implies that the error in estimating the PDF from a limited sample gets smaller as the sample size increases. The estimated PDF (the class estimator) collapses on the unknown true PDF as more patterns in the sample become available.

An example of the Parzen estimation of the PDFs is given below for the special case that the multivariate kernel is a product of the univariate kernels. In the case of the Gaussian kernel, the multivariate estimates can be expressed as :

$$f_k(X) = \frac{1}{(2\pi)^{p/2} \sigma^p} \frac{1}{m} \sum_{i=1}^m \exp \left[ -\frac{(X - X_{ki})^T (X - X_{ki})}{2\sigma^2} \right]$$

where :

- k = class or category
- i = pattern number
- m = total number of training patterns
- $X_{ki}$  = ith training pattern from category or population  $\pi_k$
- $\sigma$  = smoothing parameter
- p = dimensionality of feature (input) space.

The estimated PDF for a given class, say  $f_k(x)$  is the sum of small multivariate Gaussian distributions centered at each training sample. However, the sum is not limited to being Gaussian. It can in fact approximate any smooth density function. The smoothing factor  $\sigma$  can alter the resulting PDF. Larger  $\sigma$  causes a vector X to have about the same

probability of occurrence as the nearest training vector. The optimal  $\sigma$  can be easily found experimentally.

An interesting feature of the PNN approach is that the estimated PDFs can be used not only for classification but also to estimate the posterior probability that a vector  $X$  belongs to class, say  $\pi_1$ . If the classes are mutually exclusive, we have from Bayes theorem :

$$P[\pi_1 | X] = \frac{P_1 f_1(X)}{P_1 f_1(X) + P_2 f_2(X)}$$

Also the maximum of  $f_1(x)$  and  $f_2(x)$  is a measure of the density of the training samples in the vicinity of  $X$  which can be used to indicate the reliability of the classification.

For more details on the theory the reader is referred to Johnson and Wichern (1992), Specht (1990, 1996), and NeuralWare (1993).

The original neural network implementation of the above theory (Specht 1990) is shown Figure 5.3 for a 2-class problem. The input units are merely distribution units that supply the same input values to all of the pattern units. Each pattern unit forms a dot product of the input pattern vector  $X$  with the weight vector  $W_i$  such that  $Z_i = X \cdot W_i$ , and then performs a nonlinear operation on  $Z_i$  before outputting its activation level to the summation unit. Instead of the sigmoid activation function commonly used for the MLF the

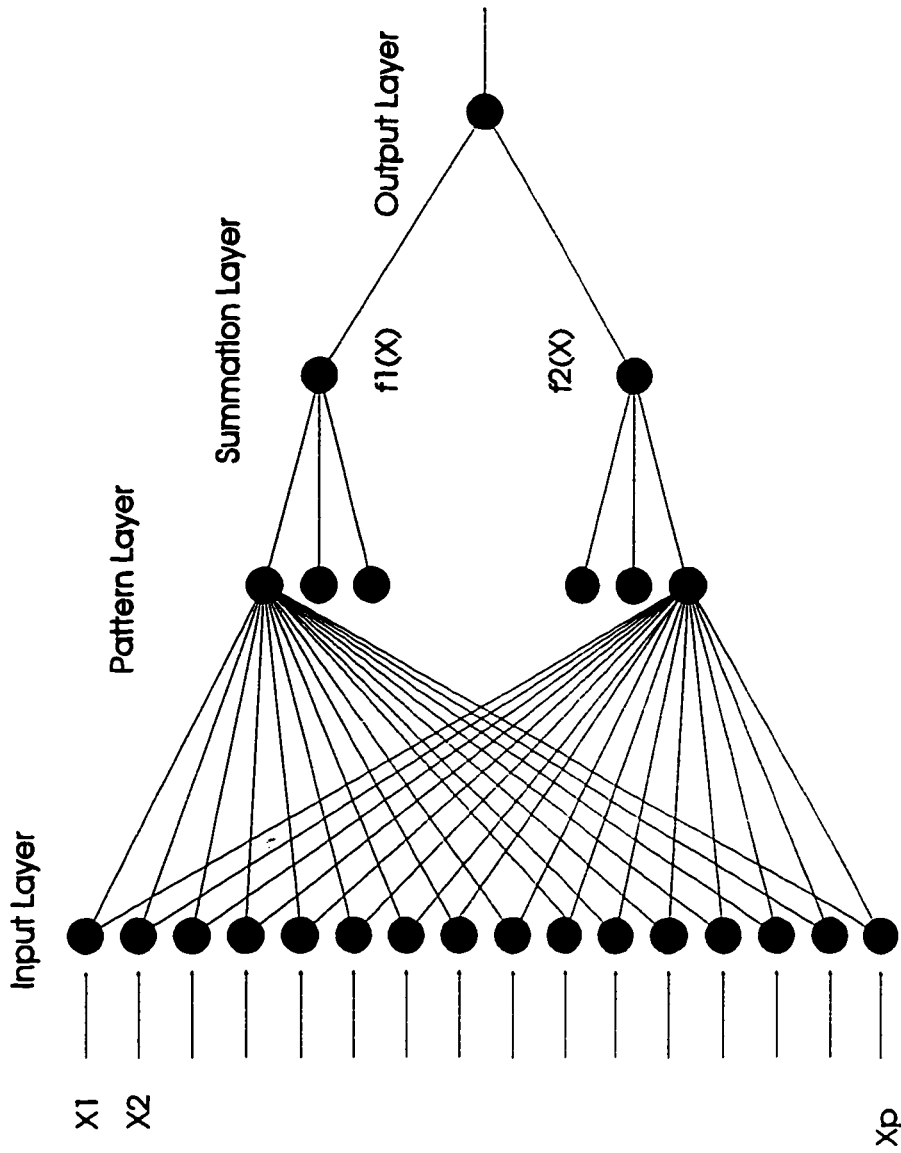


Figure 5.3. The Original Probabilistic Neural Network (PNN)



nonlinear operation used here is  $\exp[(Z_i - 1) / \sigma^2]$ . Assuming that both  $X$  and  $W_i$  are normalized to a unit length, this is equivalent to using  $\exp[-(W_i - X)^T(W_i - X) / 2\sigma^2]$  as follows:

the term

$$\exp\left[-\frac{(X - W_i)^T(X - W_i)}{2\sigma^2}\right]$$

can be rewritten as

$$\exp\left[\frac{(2X^T W_i - X^T X - W_i^T W_i)}{2\sigma^2}\right]$$

and since all inputs to the classifier have norm 1, both the dot products  $X^T X$  and  $W_i^T W_i$  equal unity and the exponential term reduces to:

$$\exp\left[\frac{(X^T W_i - 1)}{\sigma^2}\right]$$

Each summation unit simply sums the outputs from the pattern units that correspond to one of the classes. The output, or decision units are two-input neurons that produce binary outputs. They have a single variable weight  $C_k$  :

$$C_k = \{ [C(1|2) / C(2|1)] * [P_2 / P_1] * [n_1 / n_2] \}$$

where :

$n_1$  = number of training patterns for category  $\pi_1$

$n_2$  = number of training patterns for category  $\pi_2$

The network is trained by assigning a pattern unit for every training pattern, and setting the  $W_i$  weight vectors in each one of the pattern units equal to the corresponding  $X$  pattern in the training set, and then connecting the pattern unit output to the appropriate summation unit.

### 5.3.2 Advantages of the PNN-family of networks

The PNN above, as well as its variations that will be discussed shortly, have several advantages, the most important of which is that training is easy and instantaneous. Specht (1996) reports that the PNN can be used in real time because as soon as one pattern representing each category has been observed, the network can begin to generalize to new patterns. As new patterns are observed and stored in the network, the generalization improves and the decision boundary becomes more complex. The advantages of the PNN can be summarized as follows:

- Both training and re-training are easy and instantaneous,
- It can be used in real time as discussed above,

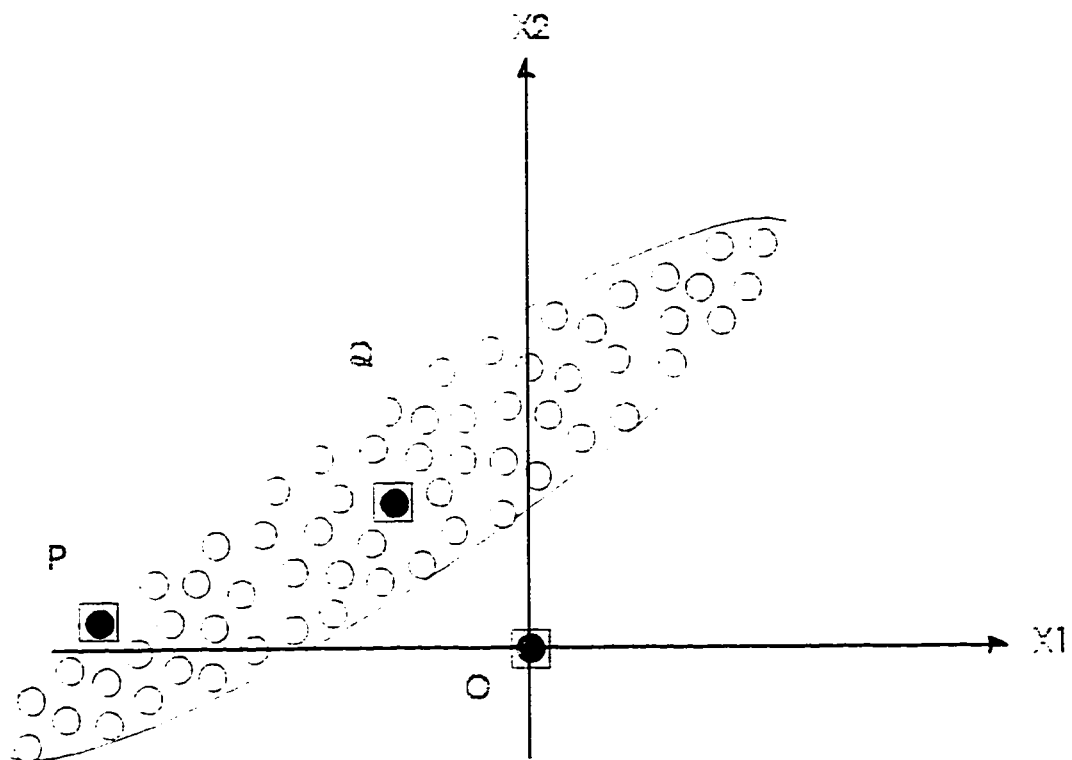
- The shape of the decision boundary can be made as complex as necessary or as simple as desired, by choosing the appropriate value of the smoothing parameter  $\sigma$ ,
- The decision surface approaches Bayes optimal,
- Erroneous samples are tolerated.
- Sparse samples are adequate for the network performance.
- $\sigma$  can be made smaller as the sample size gets larger.
- For time varying statistics (or statistics varying from site to site), old patterns can be overwritten with new patterns, and
- The cost of misclassification could be varied with time as the relative importance of detection versus false alarms changes with time of the day and location of the freeway section relative to the network.

One more important advantage is the possible chip implementation of a PNN-based AID framework. Unlike many other networks such as the MLF, the PNN operates entirely in parallel, without the need for feedback from individual neurons to the preceding layer of neurons. For systems involving thousands of neurons that can not fit in a single semiconductor chip, such feedback paths could quickly exceed the number of pins available on the chip. With the PNN, any number of chips could be connected in parallel to the

same inputs, a design that has been already implemented (Specht 1996). The important implication of this is the possibility of incorporating real parallel processor incident detection chips in a road-side cabinet. All the necessary processing could hence be done on site and only the resulting detection output would need to be transferred back to TMC in charge for further action. This adds important flexibility to the incorporation of incident detection into any intelligent Transportation System (ITS).

### **5.3.3 Statistical Distance and The Proposed Modified PNN2**

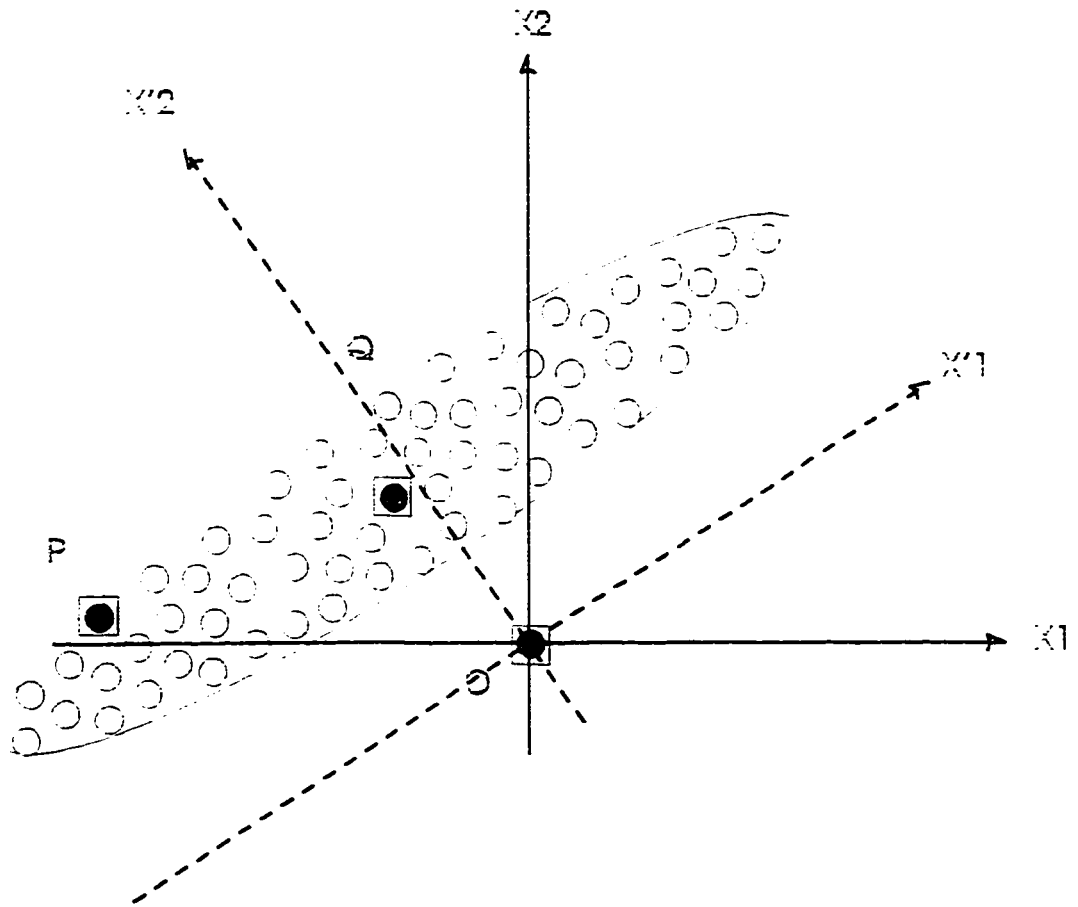
The PNN described above uses a symmetric PDF estimation kernel. This symmetric kernel is the result of using a single smoothing parameter  $\sigma$  that applies equally to all input dimensions, and using Euclidean distance as a measure of nearness of the different patterns. The use of such a symmetric kernel is inappropriate for many applications because it doesn't account for differences in variations along the axes nor the presence of correlation among the variables constituting the pattern vector. This is illustrated heuristically in Figure 5.4 where the use of Euclidean distance results in erroneous classification. In this figure, the Euclidean distance from a point such as P to the center of the cluster Q is larger than the Euclidean distance from the origin O to Q. However, P appears to belong to the cluster more than O does (Johnson and Wichern, 1992).



**Figure 5.4. Improper Use of Euclidean Distance**

Several approaches to solving this problem have been proposed in the literature. Specht (1992, 1994, and 1996) used a different smoothing parameter for each input dimension. Traven (1991) and Streit (1990 and 1994) estimated the PDFs of the different classes as a sum of gaussian kernels with a full covariance matrix. All these approaches have in common an iterative training procedure similar to that used by the MLF. Although performance improves, the training process is no longer instantaneous and is also prone to getting trapped in local minima, in the same way as the MLF is. Very recent training procedures using genetic algorithms overcome the local minima problem and are faster than conventional training approaches. However, genetic-based training is still not instantaneous. Potential benefits from using genetic algorithms will be explored in due course in this study.

In this research, a new plausible possibility to solve this problem using statistical distance instead of Euclidean distance is proposed. Statistical distance is a nearness measure that accounts for the different variability of each input variable as well the interdependence or correlation of these variables. Details of this distance measure can be found in Johnson and Wichern (1992). Using statistical distance is equivalent to rotating the axes of the input space in such away that input variables appear uncorrelated as in Figure 5.5, and then dividing each new dimension by its new variance in the rotated space. The outcome of this transformation is a new input space in which the use of symmetric kernels and Euclidean distance is appropriate.



**Figure 5.5** Axes Rotation for Statistical Distance

Two questions remain though. First, what kind of rotation would yield a set of uncorrelated axes? Second, how to modify the PNN to account for this distance measure?

The answer to the first question is to use the well known principal components in place of the original variables. Algebraically, principal components are particular linear combinations of the original set of random variables. Geometrically, these linear combinations represent a new coordinate system of axes by rotating the original ones. The new rotated linear combinations are uncorrelated and their variances are maximized. The following result gives the principal components given the original variables and their covariance matrix (Johnson and Wichern, 1992):

Let  $\Sigma$  be the covariance matrix associated with the random vector  $\mathbf{X}$ ;

Let  $\Sigma$  have the eigenvalue - eigenvector pairs

$$(\lambda_1, e_1), (\lambda_2, e_2), \dots, (\lambda_p, e_p)$$

where

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p \geq 0;$$

The  $i$ th principal component is given by

$$Y_i = e_i' X = e_{i1} X_1 + e_{i2} X_2 + \dots + e_{ip} X_p \quad i = 1, 2, \dots, p$$



and

$$\text{Var}(Y_i) = e_i' \Sigma e_i = \lambda_i \quad i=1,2,\dots,p$$

$$\text{Cov}(Y_i, Y_k) = e_i' \Sigma e_k = 0$$

The original input vector  $\mathbf{X}$  is transformed into the rotated vector  $\mathbf{Y}$  using the above relations. It should be emphasized that the original input vector is unchanged. Instead, the coordinate system used for describing the vector is changed. The component variables of the vector in terms of the rotated axes are then divided by their standard deviations  $\sqrt{\lambda_i}$  to equalize the variances.

Figure 5.6 shows the proposed modified version of the PNN (referred to as PNN2) that takes the above transformations into account. The previous input layer of the PNN is replaced by two layers: an input layer and a transformation layer. The weights between the input layer and the transformation layer are the eigenvectors of the training sample covariance matrix. The transfer function in the units of the transformation layer simply divides the weighted input to the unit by the standard deviations  $\sqrt{\lambda_i}$ . Beyond this layer, computations are identical to those of the original PNN described before, only using the principal components instead of the original variables. Using these modifications, the new network accounts for the effects of correlation and widely varying variances without losing its instantaneous training edge as it still uses symmetric kernels.

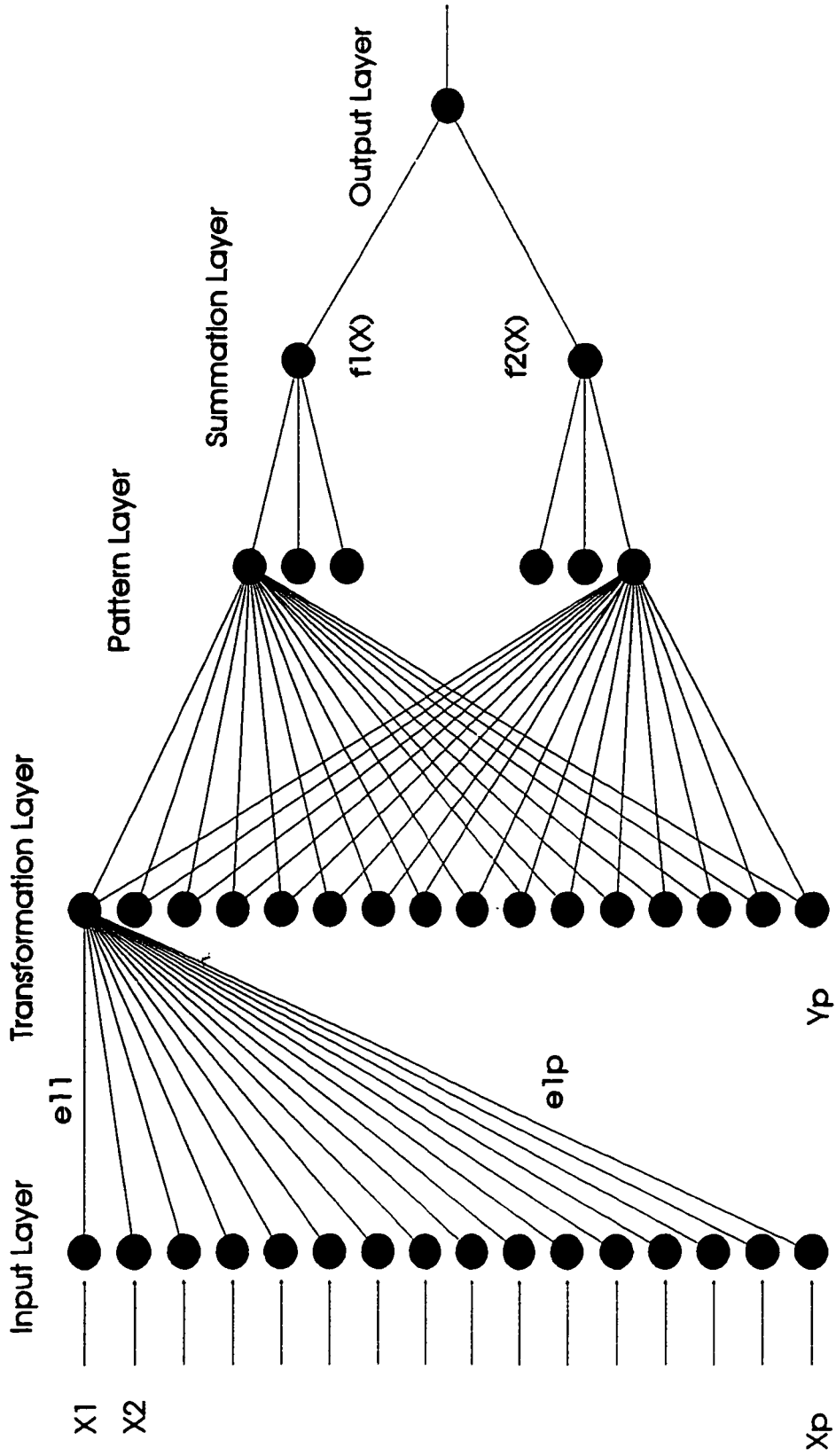


Figure 5.6. The Modified Probabilistic Neural Network (PNN2)

#### 5.4 PRELIMINARY ANALYSIS OF THE LOGICON AND THE PNN NETWORKS FOR INCIDENT DETECTION

The purpose of this section of the study is to train and test the proposed networks, compare their performance relative to each other as well as relative to the performance of the MLF by Cheu (1994). Several other conventional algorithms were previously calibrated and tested using simulated incident data, but their performances were reported to be inferior to that of the MLF (Cheu 1994). Hence the performance of the MLF is used here as an upper-end bench mark. To facilitate comparative analysis, the study site, the simulated data set from the SR-91 freeway, and the performance measures are kept exactly the same in this portion of the study as well. The expected outcome from the comparative analysis in this phase is the nomination of the most promising architecture which would be the core of the proposed universal framework and the subject of further in-depth investigation.

The Logicon, the PNN, and the modified PNN2 were implemented using the simulation data previously described in chapter 3. Each network was trained using a portion of the simulation data set, titled set 1. Another portion of the data, titled set 2 was used to monitor the training progress during any training session. A third portion of the data, titled set3, was used for the final testing and evaluation of the trained networks.

### 5.4.1 Training the Logicon Network

The configuration design and training procedure for the Logicon network is more sophisticated than both the MLF and the much simpler PNN. The following sections describe the key steps underlying the training procedure.

#### 5.4.1.1 Initialization and Choice of Prototypes

The projection operation in the Logicon architecture allows not only for the formation of open and closed boundaries but also for a good near-optimal setting of the weights and thresholds of the network. The latter is the key to reducing the training time of the network. The maximum possible value of the activation input to any hidden layer node occurs when  $W' = X'$  for any given threshold value. Therefore, for classification problems, if the weight vector  $W'$  of a node in the hidden layer is set equal to some input  $X'$  of class  $c$ , then the output of that node will be maximum when the input is  $X'$ . That node becomes a hyperspherical prototype of class  $c$  with radius  $S/R$ , where  $S$  is the threshold and  $R$  is the Radius of the hypersphere. The prototype node gives an output greater than 0.5 for input points that fall inside the prototype boundary and gives its maximum output when  $X'$  is the input. The threshold  $S$  can then be varied during the training process such that the projection of the boundary back on to the original input space is some desired

open or closed region. Thus, the weights and thresholds are set according to the desired positions and radii of the prototype. After the locations and radii of the prototypes are chosen in the original input space, the weights are set in the projected  $(N+1)$  space such that the projection of the prototype back onto the original surface will be a hypersphere centered about the chosen prototype.

Obviously the choice of the prototype locations is a key step in the training process. Representative prototypes can be chosen either at random from the training set or hand-selected by the network trainer, if desirable locations are obvious. For the purpose of incident detection, the high overlap between the classes of patterns hinders such a hand-selection process and hence random choice was adopted. The radius of each prototype is determined separately by averaging the distance between the prototype and its two nearest neighbors and then setting the radius equal to half of that distance.

With a sufficient number of such prototypes in the hidden layer corresponding to representative examples of the input vectors of each class, the input space can be adequately covered and a good initial partitioning or classification attained. Clearly, this initial setting of the weight vectors is much better than a random guess and can be very close to a desirable minimum of the error function. The subsequent training process translates, expands, and shrinks the different hyperspherical prototypes to further

reduce the error. Through gradient descent learning, similar to that of the MLF, the error function is minimized.

The above process yields a properly initialized set of weight vectors between the input and the hidden layer. The weight vectors between the prototypes in the hidden layer and the output layer need to properly initialized as well to avoid negating the benefits from properly initializing the previous layer of weights. For this purpose, a matrix inversion method (NeuralWare 1993) was adopted. In this method, as the input vectors corresponding to each prototype are presented, each output neuron receives a signal from every prototype in the hidden layer. Each input has a corresponding desired output. The solution for each weight is obtained by simultaneously solving a set of equations, where each equation requires that the output node yields the desired output for each prototype. This is equivalent to a matrix inversion.

With such an initialization, the Logicon starts much closer to an optimal solution than the MLF would, and hence saves orders of magnitude in the training time.

#### *5.4.1.2 Setting the Learning Parameters*

Although the Logicon network is trained using a back-propagation procedure like the MLF, it requires much smaller learning and momentum rates. This is because the weights and thresholds are initialized much closer to a desired minimum. The appropriate rates for a Logicon network should be an order of magnitude smaller than the typical values used for the MLF. If too large learning and momentum rates are chosen, the weights change too quickly. This leads to loss of the initial good weight setting and results in a rapid increase in network error. When such a problem occurs, the Logicon becomes no different from the conventional MLF network. Another possible consequence of large rates is the instability possibility, that is, the prototype boundaries tend to oscillate rather than settle down to the desired solution. Too small rates are not desirable either because of the associated excessively slow training. In this research the learning rate was changed from 0.005 to 0.05 and the best results obtained with a 0.01 learning rate. The momentum rate was kept five times as large at a 0.05 value.

#### *5.4.1.3 Designing the Hidden Layer and Training Strategy*

It is difficult to determine in advance the number of processing elements in the hidden layer. This, coupled with slow learning can lead to very time

consuming trials to achieve the optimal architecture. Another problem that can occur with back-propagation and associated networks such as the MLF is the problem of over-training. A symptom of this is when the network is performing well on the training data, but poorly on independent test data. Overtraining occurs when the network memorizes the peculiarities of the training set. Using NeuralWare in this research for training the Logicon, the over-training problem was solved. The idea was to start with a relatively large number of hidden units, judged to be two hundreds in our particular case, and incrementally train, test, and prune the unused connections and processing elements as follows :

1. The network is initially tested to establish a "reference" level of performance,
2. Trained for a chosen number of cycles,
3. Tested. (If the performance is better than the current "best" it is saved and the re-try counter set to zero),
4. If the retry counter is less than a preset limit, repeat 2 & 3 above,
5. Reload the last best network. If there is none (network was never saved during this run), exit.



6. Test the network on both the training and test data sets,
7. For each processing element in the hidden layer : (a) disable the processing element setting its output to 1.0, (b) test the network on both training and test sets, if the performance degrades beyond a "tolerance limit", add this processing element to candidate prune list, (c) set the output of the processing element to -1.0, (d) test the network on both training and test sets, if the performance degrades beyond a "tolerance limit", add this processing element to the candidate prune list, (e) re-enable the processing element and proceed with the next one,
8. Sort the candidate prune list into order based on performance on the test set,
9. Select the top "n" units and prune them. If none, exit training,
10. Go to step 2.

Such a training process not only overcomes the problem of overtraining but also yields an optimal set of hidden units. When the error reaches a minima and starts to rebound, the weight set corresponding to the minimal error is saved as the best attainable. It was also found that the best performance was achieved when the number of hidden units was between 20 and 50. Below that, the performance degraded and above that the extra units did not

improve performance and were hence pruned. Training time was found to be about half of that of the MLF on the same machine. Most of the error reduction occurred fast at the beginning, followed by near-flat improvements.

#### 5.4.2 Training the PNN Networks

Training of both versions, namely the PNN and PNN2, is much simpler than the case for the MLF or the Logicon. The PNN needs only one pass through the training data to learn, while the PNN2 needs to go through the data twice. Training the PNN involves the following steps:

1. Normalize all vectors to norm 1.
2. Assign a pattern unit to each training exemplar.
3. Set the weight vector between the input layer and every pattern unit in the pattern layer equal to one training exemplar.
4. Set the transfer function in the pattern units to  $\exp[(Z_i - 1) / \sigma^2]$ .
5. Set the weight vectors between the pattern units and the summation unit for class  $k$  equal to  $1/m_k$ .
6. Assign the cost of misclassification and prior probabilities to the weights between the summation units and the output units.

7. Use the test data set to choose the best smoothing parameter  $\sigma$  by trying a range of values and choosing the one that yields minimal classification error.

At this stage the cost of misclassification ratio and the prior probability ratio were both set to unity. Also, the range of  $\sigma$  within which the performance peaked was found to be between 0.065 and 0.2 with the best performance attained at  $\sigma = 0.068$ .

Training of the PNN2 is similar except that it uses transformed patterns. The transformation stage requires one pass through the training data to compute the covariance matrix, and to obtain both the eigenvalues and eigenvectors of this matrix. Once they are obtained, the eigenvectors are assigned to the weights between the input and the transformation layer and the reciprocal of the square root of the eigenvalues assigned to the transfer functions in the transformation units. After the transformation stage is completed, training proceeds in the exact same way as the PNN.

It is important to emphasize that many of the key parameters of the PNN family of networks can be changed after training without the need for retraining. For instance, if the training data is sparse, one can start with a relatively large smoothing parameter, and as more data become available, the smoothing parameter could be made smaller accordingly. Also if the cost

of misclassification and/or the prior probabilities vary during operation, these can be modified accordingly without disturbing the network.

### **5.4.3 Incident Detection Performance**

After the training phase of the Logicon, PNN and PNN2 networks was completed, all three networks were tested on a common subset of the SR-91 simulation database, namely set 3, as was previously described. The DR, FAR, and average TTD were computed separately for single lane blocking incidents and for multilane blocking incidents. Persistence tests of up to three intervals were used to further reduce the FAR due to random fluctuations of traffic. A persistence test of  $n$ -intervals means that an incident alarm is declared only if the incident condition is found to persist for  $(n+1)$  consecutive applications of the algorithm. Table 5.1 categorizes the performance measures for one lane and multi-lane blocking incidents on data set 3. The measures computed from these two categories were combined using weights of 0.9 and 0.1 respectively to produce a final weighted DR, FAR, and average TTD. This was based on a previous study in Los Angeles which found that for lane blocking incidents, 90% were one lane blocking and 10% were multilane blocking (see Guiliano 1989, Cheu 1994). The performance envelopes (DR versus FAR using different persistence tests) are shown in Figure 5.7 for the Logicon, PNN, PNN2 and the MLF networks.

		One-lane-blocking			Two-lane-blocking			Combined		
Net.	Pers.	DR	FAR	TTD	DR	FAR	TTD	DR	FAR	TTD
Logicon	0	98.2	4.75	49	99.8	3.58	38	98.3	4.63	48
	1	94.1	2.62	97	99.4	1.80	72	94.6	2.54	95
	2	89.4	1.79	145	99.0	0.99	102	90.4	1.71	141
	3	78.8	1.23	188	98.8	0.64	133	80.8	1.17	183
PNN	0	97.0	14.00	35	100.0	3.00	70	97.3	12.9	39
	1	72.5	2.40	147	97.6	0.96	105	75.0	2.26	143
	2	67.0	1.60	176	96.3	0.34	137	70.0	1.47	172
	3	55.5	0.84	222	95.9	0.14	173	59.5	0.77	217
PNN2	0	100.0	4.80	73	100.0	4.50	44	100.0	4.77	70
	1	95.1	2.68	172	99.8	1.79	88	95.6	2.59	163
	2	83.7	1.30	229	98.4	0.88	121	85.1	1.26	218
	3	68.4	0.66	260	98.2	0.50	153	71.4	0.64	249
MLF	0	78.0	1.50	206	97.0	1.06	87	79.7	1.46	194
	1	65.0	0.44	252	96.0	0.20	114	68.1	0.42	238
	2	56.0	0.22	287	95.0	0.16	146	59.9	0.21	273
	3	46.0	0.18	311	93.0	0.10	175	50.7	0.17	297

**Table 5.1. Testing the Logicon, PNN, PNN2 and the MLF on Data Set 3.**

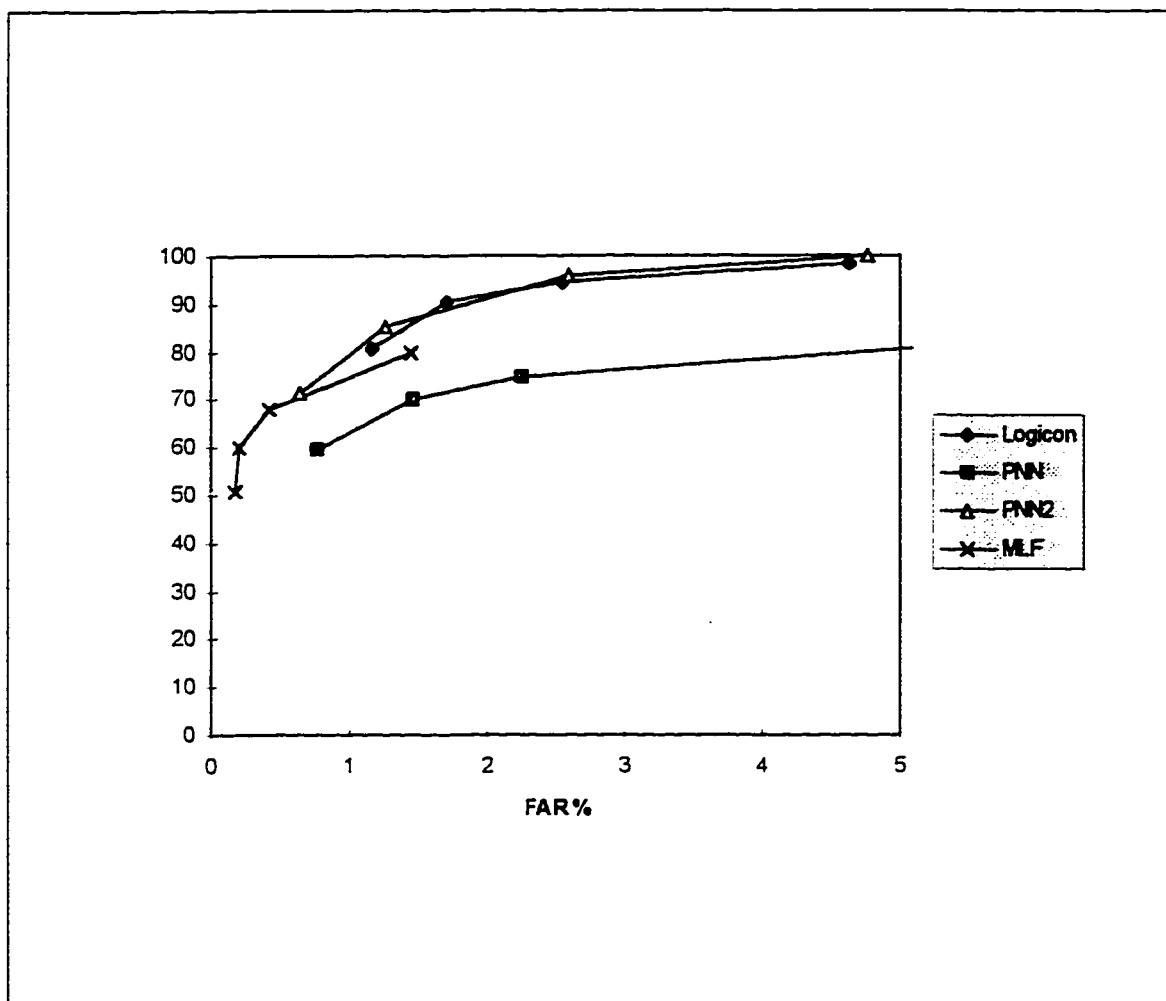


Figure 5.7 Logicon, PNN, PNN2 and MLF Performance on data set 3.

#### 5.4.4. Comparative Evaluation and Preliminary Conclusions

Careful investigation of the detection results in Table 5.1 and Figure 5.7 shows that:

1. The performance of the Logicon network is slightly better than the MLF. However the improvement appears to be insignificant on detection performance grounds alone. The speed advantage still gives the Logicon an edge over the MLF.
2. The performance of the original PNN is lower than that of the MLF.
3. Using the modified PNN2 which incorporates the statistical distance metric via principal component transformation, the performance improves and becomes competitive with the Logicon and the MLF. The instantaneous training of the PNN2 makes it the most promising amongst the three architectures.

Table 5.2 below shows a universality comparison of the Logicon, PNN2, and the MLF. All three networks are comparable in terms of performance. Training complexity of the Logicon is even worse than that of the MLF. It has more very sensitive parameters to be hand-crafted, and prototypes to be selected, all of which require higher developer skills. In contrast, training of the PNN family is very simple and robust. Principal components are unique and require no hand-crafting of any kind. The smoothing parameter  $\sigma$  is a

Universality Attribute	Algorithm		
	MLF	Logi.	PNN2
1. Highest performance.	√	√	√
2. Fast, robust and automated training and retraining	x	x	√
3. Reasonable TMC implementation requirements	x	x	√
4. Transferable logic:	√	√	√
5. Reasonably transferable training / calibration parameters	Not tested yet		
6. Minimal initial training data requirements	x	x	√
7. Account for prior probabilities of incidents	x	x	√
8. Account for the unequal costs of misclassifying traffic patterns	x	x	√
9. Capable of producing the posterior probability of an incident	%	%	√
10. Estimate incident severity	%	%	√
11. Capture incident duration	%	%	%
12. Statistical and theoretical soundness and clarity	x	x	√
13. Immunity to minor traffic fluctuations effects	√	√	√
14. Immunity to bottleneck effects	x	x	x
15. Immunity to consistent loop detector biases	x	x	x

x: absent attribute

v: fulfilled attribute

%: attribute fulfilled to some extent, but not fully.

N.A.: Not Applicable.

**Table 5.2. Universality Comparison of the Logicon, PNN2 and the MLF**



single parameter, the choice of which is not confusing and it does not interact with other parameters in a complex non-linear fashion as is the case for the MLF and the Logicon. This simplicity and fastness of the training process make it possible to retrain the network as needed on-line in order to adapt to any changes in the freeway environment. The PNN family, as has been described earlier, can generalize even if the training set has only one exemplar from each category. Prior probabilities, cost of misclassification, and posterior probabilities are inherent components of the structure of the PNN networks. The posterior probability form of the PNN output will be capitalized upon to indicate incident severity and continual presence in the field as will be described later. The theory underlying the PNN is very well established, understood and accepted. All of these advantages makes the modified PNN more appealing than the Logicon or the MLF. It is therefore chosen for further in-depth investigation in the rest of this study. The modified PNN (hereafter referred to as simply the PNN) will be used as the core of the proposed universal incident detection framework. A pre-processor input feature extractor will be added to it that enhances its transferability and makes it immune to both bottlenecks and virtual bottlenecks, and a post-processor that translates the probabilistic outputs of the network into a continuous estimate of existence of an incident in the field, not based on a singular output but rather incorporating previous outputs into the current decision. The transferability testing and enhancement, and the pre and post

processor details will be the subject of the rest of this research. Also to be investigated is a multi smoothing parameter version of the PNN optimized by a genetic algorithm.

## **CHAPTER 6**

### **DETAILED DEVELOPMENT OF THE PNN-BASED FRAMEOWRK**

#### **6.1 INTRODUCTION**

This chapter is dedicated to an in-depth investigation of the PNN, and the development of the complementing components of the proposed universal AID framework. First, the transferability of the PNN as well as several other well known algorithms is rigorously tested using real incident and loop detector databases from the I-880 and I-35W freeways. The causes of the high overlap between the incident and normal pattern classes in the conventional input feature space are identified. A pre-processor feature extractor is presented that reduces this overlap and significantly improves transferability. The on-line learning in real-time capability of the PNN is investigated and very promising results are presented. A post-processor module that generates a continuously updated probabilistic estimate of the presence of an incident in the field is presented. This post-processor uses the probabilistic outputs from the previous application intervals as a prior-probability for the current decision and generates a Bayesian estimate of the current probability of an incident. Finally, the complete system is evaluated and its universality assessed.

## 6.2 TRANSFERABILITY TESTING AND EVALUATION

The objective of this phase is to assess the transferability of AID algorithms from the site/data on which they were initially trained to new, previously unseen, potentially different sites. As far as this research is concerned, transferability is defined as the direct application of a given trained algorithm to a new different site (or the same site after significant changes with time) *without* recalibration of any parameters. Due to the fact that new sites have different site-statistics, deterioration in the performance of any algorithm is expected, and this section is dedicated to the assessment of the extent of such deterioration, and the analysis of the underlying causes in order to develop potential transferability-improvement techniques.

The real loop data and reported incidents from the I-880 freeway in Alameda County, California and the I-35W in Minneapolis, Minnesota were used to test the developed algorithms which were trained on simulation data from the SR-91 freeway in Orange County. The loop data corresponding to the real incidents were used to assess the DR and the TTD. One Hundred hours of incident-free loop data from each site were also randomly extracted and used for FAR testing. Both the PNN and the MLF were tested on these incident and normal databases. To broaden and generalize the comparative evaluation, two other algorithms studied by Cheu (1994) were also tested, namely, the California algorithm #8 and the Minnesota (DELOS) algorithm.

It is worthy to emphasize that the four algorithms were all trained on the same simulation data set in order to enable a fair comparison on common grounds.

Table 6.1 shows the results from testing the previous four algorithms on the I-880 incident and incident free data sets. Table 6.2 shows the corresponding results from testing on the I-35W database. Up to three intervals of persistence were used where applicable.

Careful investigation of the above results reveals the following important points:

1. Based on the acceptable performance limits established earlier from TMC surveys, none of the algorithms showed acceptable performance or even close to acceptable performance, either in the I-880 site or the I-35W site.

The nature of the incidents in the I-880 database caused such obviously low performance. This was due to the fact that the upstream occupancies at the I-880 test site were consistently low and did not exceed the 40-50% range even during lane blocking incidents. On the other hand, the SR-91 data used for training showed more pronounced effects of the incidents, as the upstream occupancies were as high as 80-100%. The large difference between the site-statistics of the training SR-91 data and the testing I-880

	California Alg.			Minnesota Alg.			MLF Net.			PNN Net.		
Pers.	DR	FAR	TTD	DR	FAR	TTD	DR	FAR	TTD	DR	FAR	TTD
0	44	0	134	7	0	170	35	0	306	31	0.40	1080
1	-	-	-	-	-	-	35	0	336	29	0.02	1188
2	-	-	-	-	-	-	35	0	366	29	0	1218
3	-	-	-	-	-	-	35	0	396	27	0	902

**Table 6.1. Testing Results on the I-880 Data Set.**

	California Alg.			Minnesota Alg.			MLF Net.			PNN Net.		
Pers.	DR	FAR	TTD	DR	FAR	TTD	DR	FAR	TTD	DR	FAR	TTD
0	55.2	3.64	107	19.4	0	549	60.5	6.2	279	76.1	8.16	415
1	-	-	-	-	-	-	50.7	4.1	308	40.0	2.7	348
2	-	-	-	-	-	-	47.7	3.2	296	29.1	1.3	328
3	-	-	-	-	-	-	38.0	2.6	264	24.6	0.9	465

**Table 6.2. Testing Results on the I35W Data Set.**

1. data is the prime cause of the common low performance amongst all algorithms on the I-880 site. On the other hand, the incident patterns for I-35W were relatively closer to those of the I-880 site and therefore all algorithms showed relatively better performance on the I-35W site as opposed to the I-880 site.
2. Since the statistics of the traffic in the new testing sites are significantly different from the training site as explained in "2" above, recalibration or retraining on data from the new site is necessary for the algorithms to show acceptable performance after their transfer. However, retraining or recalibrating an incident detection algorithm for each and every new site it is transferred to is not a trivial task, but is rather prohibitive. This problem motivates the utilization of the instantaneous learning capability of the PNN class of networks to counter balance the severe drop in performance due to transferability while the network is in service in real or pseudo real time, which is discussed next.

## **6.3 IN-SERVICE PERFORMANCE IMPROVEMENT OF THE PNN**

### **6.3.1 On-Site Real-Time Retraining**

As was briefly discussed earlier, the PNN family of neural networks use the training patterns as connection weights to represent the knowledge content



of the network. Other networks, such as the MLF for instance, use the training patterns to optimize the connection weights through a search mechanism such as gradient descent. This optimization process takes time and the search or training parameters have to be set or hand-crafted by the network developer, a process that requires experience and skill. For the PNN, once the training patterns are assigned to the pattern units, the PDFs of the different classes are known and the network is ready to generalize. If the operating environment changes and hence the PDFs of the different classes also change, new patterns from the new environment can replace the older ones and therefore the new PDFs are instantly re-generated. There is no time consuming search or optimization involved. This important feature allows the PNN to adapt to changes in the operating environment as they occur, be it due to transferability to a totally new site or due to temporal changes in the same site. The problem is how to obtain data from the newly changed environment or more specifically the new freeway site? Since the performance of the PNN, as well as most other AID algorithms, deteriorates after transferability, it will detect less than 100% of the incidents that occur in the field. How much less than 100% the DR will be depends on how much different is the new site and how far or close are the estimated PDFs during initial training relative to the actual unknown PDFs from the new site. Nevertheless, the algorithm will detect some incidents. If the traffic patterns during the times of the detected incidents are used to update the knowledge

content in the pattern units of the PNN by overwriting the older patterns in real time, the performance would gradually improve as more and more incidents get detected with time in service. Updating the non-incident patterns is not a problem as normal traffic data is always available.

The philosophy presented here for training the PNN is therefore different from the conventional approach in the field of AID. In the conventional approach, the system developer attempts to collect incident and loop data that would represent all possible incident scenarios in all possible field operating conditions in order to produce a generic AID algorithm. This is driven by the fact that once the network, say the MLF, is trained there would be no room for changes. The case is different however with PNN. First the network is trained to be generic, similar to the conventional training approach, but this is used only as a starting step. Once copies of the generic network are installed to operate at different sites, each local copy would gradually change according to the specifics of the local site. As time goes on, the local copies of the trained network would become less and less generic and more and more site-specific. This overcomes the problems associated with having a network that is trained to be generic to operate on a site with peculiar needs. This also overcomes the lack of transferability problem as the network gets a second chance to adapt to the changes at the new site.

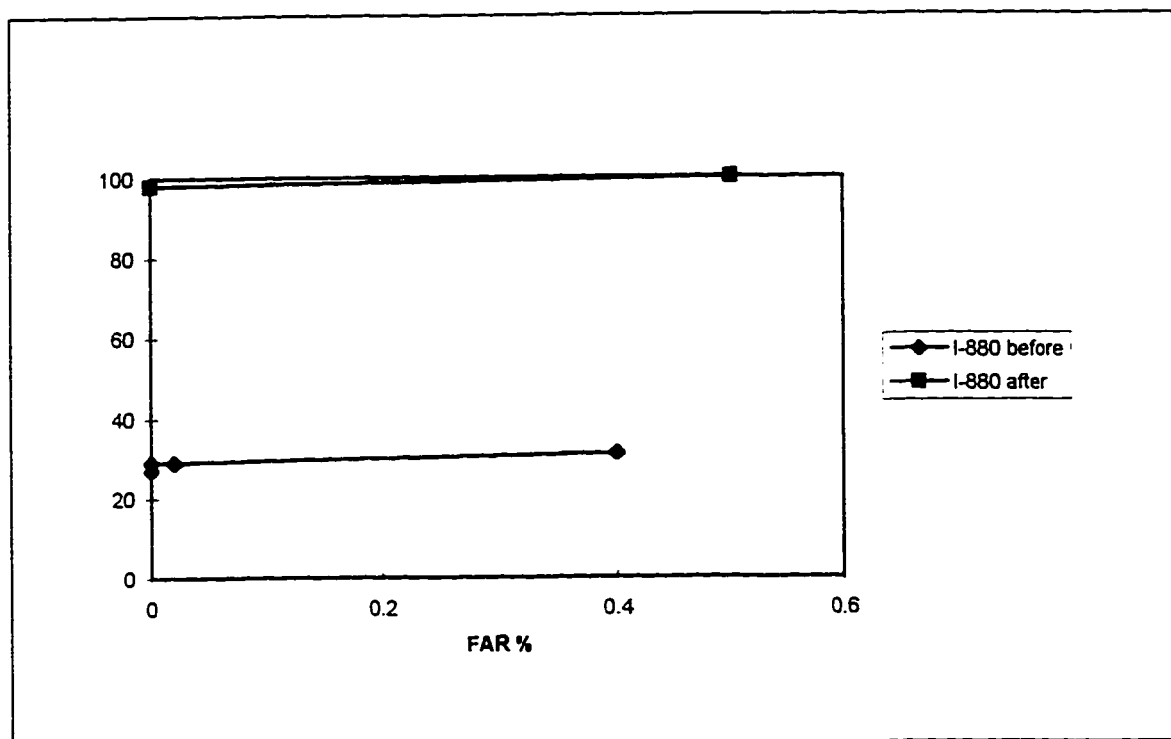
To verify this theoretical hypothesis, emulated on-site training sessions on the I-880 and the I-35W databases were conducted. Right after transferring the PNN from the simulated SR-91 freeway site to the I-880 site, the network detected about 30% of the total number of incidents as previously presented. These detected incidents, together with an equivalent amount of incident-free data were used for the updating process. Similarly, after transferring the PNN from the simulated SR-91 freeway site to the I-35W site, the network detected about 40% of the total number of incidents (at 1 persistence interval). These detected incidents, together with an equivalent amount of incident-free data were used for the updating process. One retraining session was conducted for each site. After the network was *instantly* retrained on the incident data that it managed to correctly recognize in the first round of testing, a second round of testing was performed on the entire size of each database in order to assess the improvements. As shown in Table 6.3 below for the I-880 freeway, the retraining or updating process resulted in significant improvement in the performance of the network. The DR improved from only 30% to 98% and the FAR dropped to 0%. Similar significant improvement in the TTD were also evident. Results from the I-35W site also confirmed the benefits from the real-time on-site retraining process as shown in Table 6.4. Figures 6.1 and 6.2 show the before and after performance envelopes for the I-880 and I-35W sites respectively.

PNN						
Before On-Site Patterns Update				After On-Site Patterns Update		
Persistence	DR	FAR	TTD	DR	FAR	TTD
0	31	0.4	1080	100	0.5	15
1	29	0.02	1188	98	0	79
2	29	0	1218	98	0	112
3	27	0	902	98	0	142

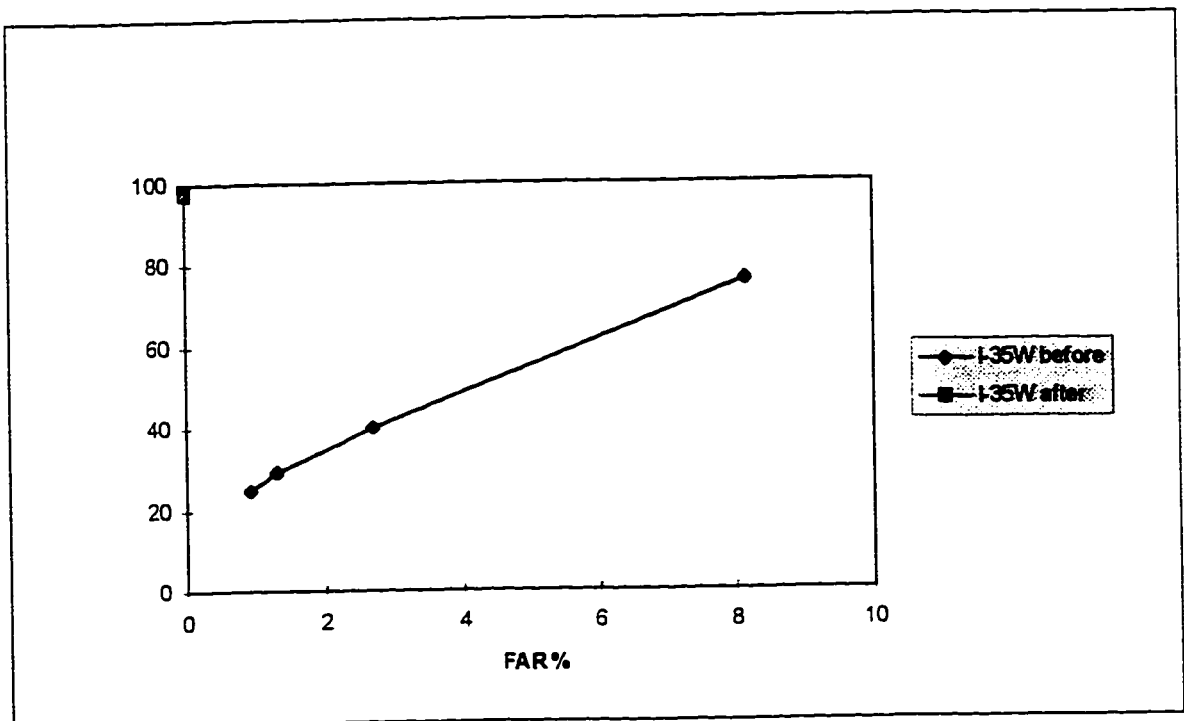
**Table 6.3. Performance Improvements After Real-Time Updating on the I-880 Patterns.**

PNN2						
Before On-Site Patterns Update				After On-Site Patterns Update		
Persistence	DR	FAR	TTD	DR	FAR	TTD
0	76.12	8.16	415	98.51	0.011	48
1	40.0	2.7	348	98.51	0.009	85
2	29.1	1.3	328	98.51	0.006	116
3	24.6	0.9	465	97.76	0.003	147

**Table 6.4. Performance Improvements After Real-Time Updating of the I35W Patterns.**



**Figure 6.1 Performance Envelopes Before and After Real-Time Updating on the I-880 Patterns.**



**Figure 6.2 Performance Envelopes Before and After Real-Time Updating on the I-35W Patterns.**

### 6.3.2 Discussion

Given the significant improvement from the real-time on-site retraining, several important points should be noted:

1. Taking the I-880 site as an example, the amount of incident data needed from the new site was less than one third of the number of incidents in a two months period of data collection. Thus, had the updating process been implemented on-line, the network would have taken only two months to improve its performance from a totally unacceptable level of performance to almost an ideal performance of 100% DR and 0% FAR. Dividing the amount of improvement in the DR by the number of weeks in two months indicates that the network improves at a fairly high rate of about 8.5% every week in service. Even better rates are obtained from the I-35W site.
2. Testing the algorithm on the I-880 data set is a fairly harsh test since the effects of the incidents on traffic conditions were not severe, resulting in occupancies below 50%. In an average case, the drop in performance due to transferability is expected to be less than the case of the I-880 test. Hence, the on-line time required for the PNN to achieve ideal performance would also be less than 2 months, assuming a similar rate of incident occurrence. The I-35W site is an example of such a case where the performance drop after transferability was not as bad as in the case for the I-880 site. A higher



percentage of incidents were initially detected, which allows for faster recovery of performance.

3. No other known algorithm or network can achieve similar *on-line* improvement in performance without close attendance of a system developer. This is due to the nature of the training process of the PNN detailed previously. Unlike the PNN, the MLF uses the training patterns to develop connection weights using an error minimization technique. When new patterns become available at the new site, the network has to be retrained off-line, by the user TMC, to reach a new set of weights that minimize the error, given the new conditions. In fact, the MLF, when trained off-line on the same data, achieved similar optimal performance. However, it is the process of off-line re-training or re-calibration at each and every new site that we are trying to avoid so long as there is not a significant difference in the ultimate performance.

4. Since false alarms are possible, one might correctly argue that the detected incidents should not be used directly to update the training patterns without incident verification. If incident verification is a necessary step then the updating process could only take place in pseudo-real time, delayed by the amount of time required to verify the truth of an incident alarm, and conditioned upon an "update permission" by a TMC operator, which may be an effortless simple press of a button. However, it might be possible to avoid

conditioning the update process on incident verification. This could be achieved by utilizing the “cost of misclassification ratio” discussed earlier which could be set in such a way that a 0% FAR is favored over a high DR. This way, the network would have a 0% FAR right after transferability at the expense of an initial low DR. As the network starts to detect a few *true* incidents and update its weights the DR would start to improve gradually with time in service.

## **6.4 ENHANCED FEATURE EXTRACTION FOR UNIVERSAL TRANSFERABILITY**

### **6.4.1 Causes of the Lack of Transferability Problem**

Incident detection is essentially a pattern recognition problem. Loop-detector-based traffic data, such as volume and occupancy, over several minutes, are usually used as an input vector fed directly into the detection algorithm. The dimensionality of the input vector defines the dimensionality of the feature space of the problem. The role of the detection algorithm is to partition the input feature space into two classes, one for incident-related vectors, and the other is for incident-free vectors. Unfortunately, the two classes of patterns are highly overlapping in the feature space. Figure 6.3.a & 6.3.b illustrate the extent over which each class of patterns extends in a

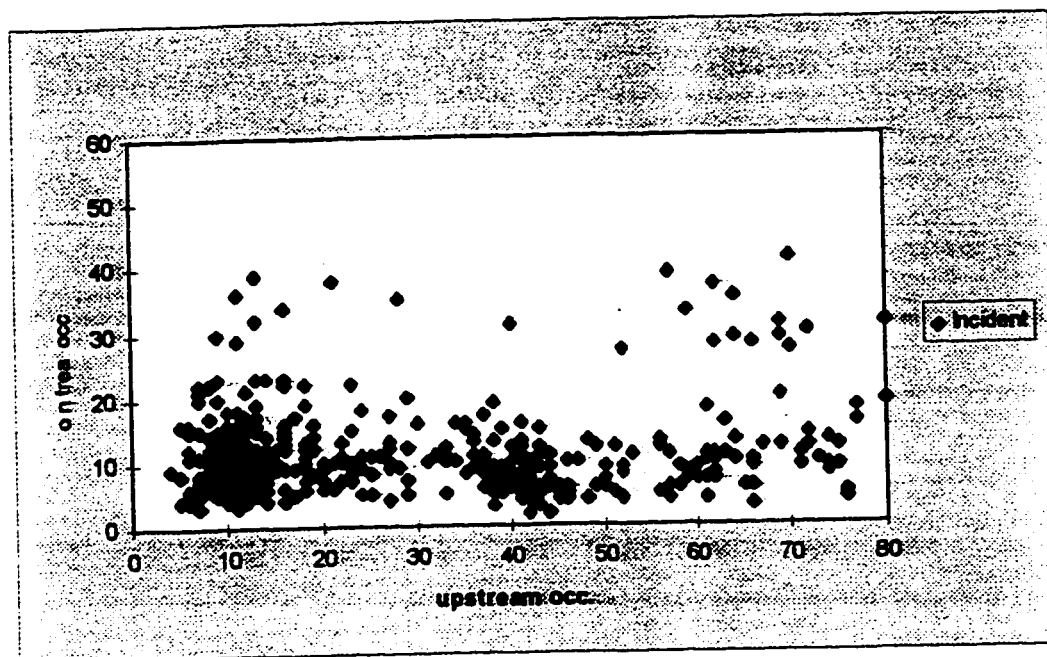


Figure 6.3.a Incident Data in Two Dimensional Input Space

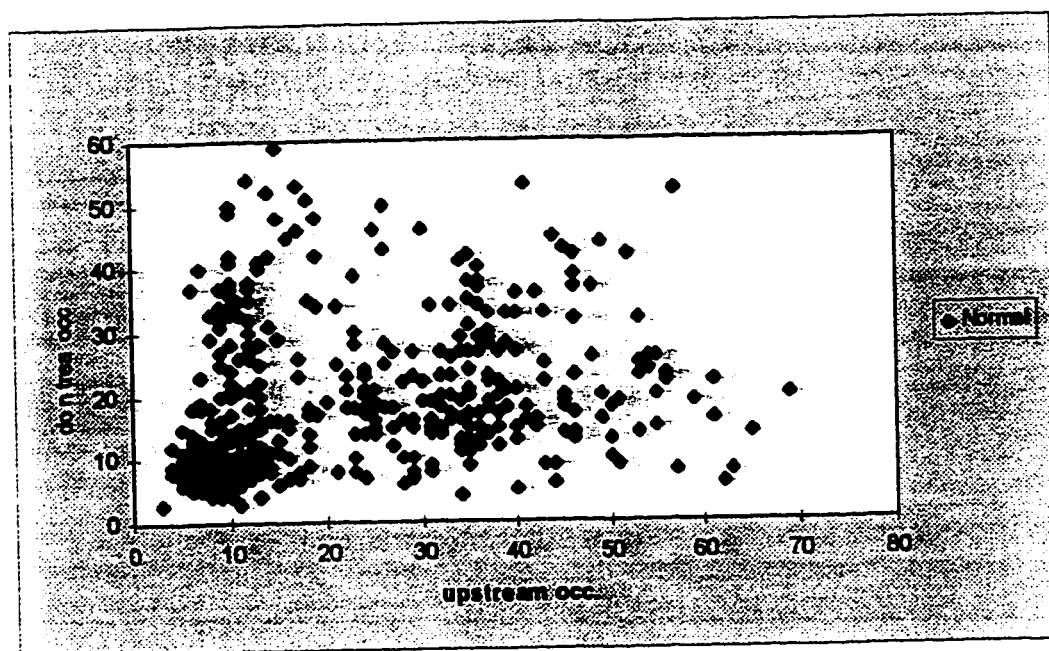


Figure 6.3.b Normal Data in Two Dimensional Input Space

simplified two dimensional input space. This simplified input space is composed of upstream and downstream occupancies from a test section from the SR-91 freeway in Orange County. Comparison of the figures (plotted separately for clarity) clearly shows the high overlap extent between the two classes. Such an overlap is believed to be attributed, at least in part, to the following reasons:

- The shockwaves associated with each incident may take a few minutes to travel from the actual incident location to the loop detector locations. The boundary of the congested region for instance propagates upstream at an approximate speed of 10 miles per hour, where the exact value depends on the characteristics of the incident, the geometry of the freeway segment and the level of traffic at the time of the incident. On the other hand, the input patterns are usually marked as incident-related for calibration purposes immediately following the onset of the incident. During the few minutes of shockwave travel time, the input patterns at the loop detector locations are effectively normal but yet are mistakenly marked as incident-related. The result is a few points in the feature space that are marked as incident-related but actually reside well in the heart of the normal class of patterns. The opposite is also true for the clearance time of an incident. Once the incident is physically removed, and before the clearing shockwave reaches the detector locations, the input vectors are marked as normal (or non-incident) while they in fact reside well in the

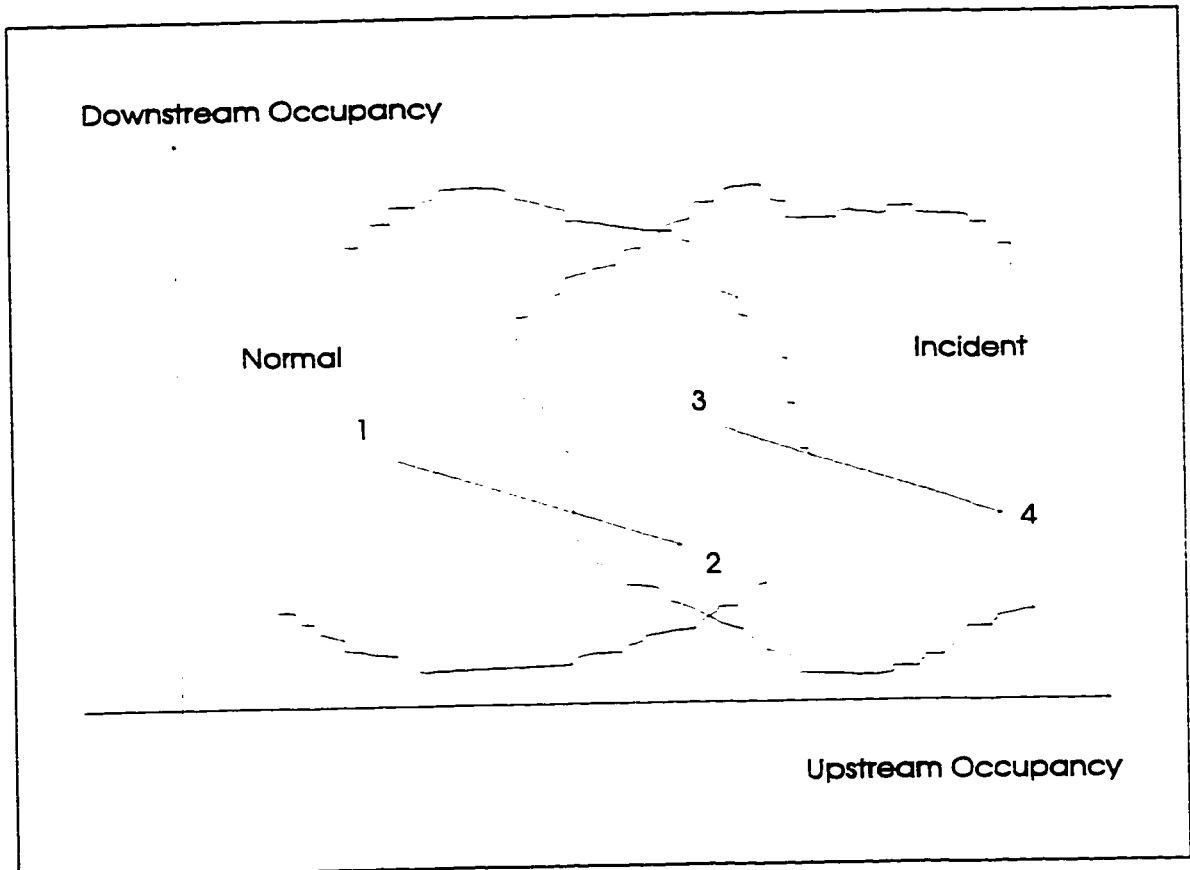
heart of the incident class of patterns. Such an overlap confuses any AID algorithm while being trained to separate the two classes of traffic patterns.

- The pre-incident conditions vary from one location to another and from time to time at a given location. Consequently, the location of the incident-free patterns varies widely and covers most of the input space. On the other hand, and for the same pre-incident conditions, the extent to which an incident affects the traffic conditions and hence the extent to which the data points migrate in the feature space vary not only with the characteristics of the incident but with the characteristics of the freeway as well. Consequently the location of the incident patterns vary widely as well and covers a wide range of the input space. The two widely dispersed classes of patterns hence overlap and ambiguously share most of the feature space. Figure 6.4 is a schematic illustration of such a problem where point # 2 represents an incident condition that still resides in the normal region and point # 3 is a normal point that is well inside the incident region.
- Sudden geometry changes on a freeway such as lane drops, bottlenecks, or major on or off ramps create a transitional section on the freeway with upstream occupancy and volume levels that are different from those at the downstream side, even under incident-free conditions. For instance, if

the capacity of a given bottle neck is exceeded during rush hours, the occupancy levels on the upstream side far exceed those at the downstream side of the section, a situation that is incident-free but yet resembles the traffic patterns associated with incidents. The result is a set of incident-free input vectors that reside within the incident-related class causing false alarms. It is worthy to emphasize that incidents are nothing but non-recurring capacity reducing events, i.e. bottlenecks.

- Some virtual bottleneck situations similar to the above could also arise due to loop-detector biases. For instance, if the upstream loop detector consistently reports higher occupancies than the downstream station under normal incident-free conditions, the detection algorithm will consistently generate false alarms at this location.

The above sources, combined, produce highly overlapping classes of traffic patterns. Even worse, any boundary that gets generated by any classifier will be location-dependent. In cases of transferring an already trained algorithm to a new site, a process of re-calibration on local data might therefore be essential. Although the PNN-based framework makes such retraining simpler, it would be beneficial to try to enhance the transferability to reduce the performance deterioration and to make role of the retraining part of the overall process easier. The following section describes the details of a transferability-enhancing input feature extraction process.



**Figure 6.4. Varying Pre-incident Conditions and Post-incident Effects Contribute to Classes Overlap.**

### 6.4.2 Enhanced Feature Extraction

The objective of the proposed input features is to form an input feature space in which the incident and normal pattern classes are as distinct and separate as possible. The backbone of the proposed approach is twofold: [1] replacing the volume and occupancy inputs with the volume and occupancy deviations from their Averages-for-the-Time-and-Location (ATL), and [2] beginning the marking of traffic patterns as incident-related only after the incident-related shockwaves reach the detector locations and not at the physical onset of the incident.

In the first case, a set of day-of-the-week-specific historical data files are prepared at each loop detector location that holds location-specific incident-free volume and occupancy averages over successive intervals (15 minutes for instance). As the traffic data that needs to be classified become available, and instead of feeding them directly into the detection algorithm, the historical ATLs are first subtracted from the on-line values and only the deviations from these ATLs are fed into the algorithm. These new features have the following set of advantages:

1. Unify the pre-incident conditions at all locations and at all times. In the input feature space, all the pre-incident patterns will be shifted to fall around the origin. Consequently, all incident-free situations will look the



same regardless of any local traffic pattern effects. Incident-related patterns will evidently fall far from the origin of the feature space.

2. Sudden changes in the geometry of the freeway such as lane drops, bottle necks, or major on or off ramps will have no effect on the location of the incident-free patterns in the feature space. For instance, if the capacity of a bottle neck is usually exceeded during rush hours, i.e. the incident-free occupancy levels on the upstream side far exceed those at the downstream side of the section during this period, this will be reflected in the historical ATLS file for this time period. Hence, under an incident-free situation at this location and during this period of time, the deviation from the ATLS will always be practically zero. If an incident occurs, the already existing gap between the upstream and downstream occupancies for instance will widen further, and the deviations from the historical ATLS will start to be of considerable magnitude indicating the onset of an incident. This way, bottleneck situation and funneling freeway sections could be effectively, and seamlessly handled by the algorithm without any requirement of special local loop-detector configuration (examples of such requirements can be found in Lin, 1995 ).
3. Similarly, virtual bottle neck situations due to loop-detector biases could also be avoided. For instance, if the upstream loop detector consistently reports higher occupancies than the downstream station under normal

incident-free conditions, the bias will be reflected in the historical ATLs file, and hence the detection algorithm will not see any deviations from the ATLs and will not generate false alarms at this location. Only when an incident occurs, the already existing gap between the upstream and downstream occupancies for instance will widen further, and the deviations from the historical ATLs will start to be of considerable magnitude indicating the onset of an incident. Loop detectors that go bad suddenly do not fall into this category because the ATLs file will not reflect any consistent abnormalities. However, sudden generation of a set of false alarms at such a location could be indicative of an “abnormal” event. In such a case the detection algorithm will detect the malfunctioning detector itself.

Table 6.5 shows a few hypothetical illustrative numerical examples of the above features. The first freeway section is a homogeneous one (no geometry changes), with equal upstream and downstream occupancies during incident-free operation. Also the occupancy values are equal to the representative values for the given time slot as stored in the ATL files. Hence, under normal conditions, there are no deviations from the ATL values. However, as an incident occurs, the upstream and downstream occupancies start to deviate from the ATL values. The second section is also a homogeneous one, but operating at higher incident-free occupancy levels which are also equal to the location-specific and time-specific ATL values. Hence, under normal

Freeway section type	ATL u.s.	ATL d.s.	Direct Occupancies				Deviations from ATLs			
			Before incident		After incident		Before incident		After incident	
			u.s.	d.s.	u.s.	d.s.	u.s.	d.s.	u.s.	d.s.
Homogeneous.	20	20	20	20	60	10	0	0	40	-10
Homogeneous.	40	40	40	40	80	30	0	0	40	-10
Bottleneck.	40	20	40	20	80	10	0	0	40	-10

ATL: Average Occupancy for the particular Time and Location.

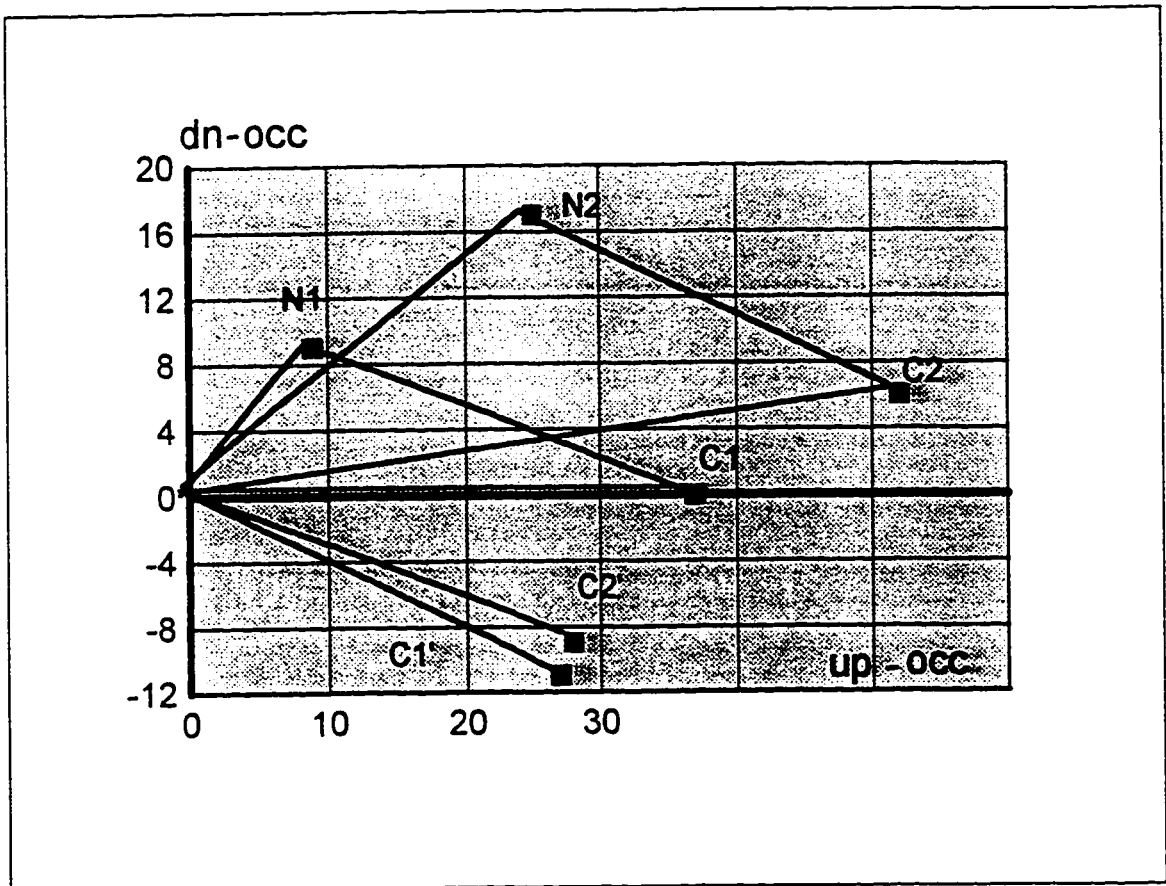
u.s.: upstream

d.s.: downstream

**Table 6.5 Examples of the New Input Features (Occupancy Values Example).**

conditions, there are no deviations from the ATL values as in the case of the previous section. However, as an incident occurs, the upstream and downstream occupancies start to deviate from the ATL values. Assuming that the incident effect is similar to the previous case, the deviations from the ATL values will be equal to those from the previous case as well. The last section represents a bottle neck situation that causes the upstream occupancies to be higher than their downstream equivalent under incident-free conditions. However, because this is usually the case at this location and at this time of the day, day of the week .. etc., the historical ATL values will reflect such an incident-free occupancy difference, and hence there will be no deviations from the ATL values. As an incident occurs, the upstream and downstream occupancies will drift further apart and will start to deviate from the ATL values. Assuming that the incident effect is similar to the previous case, the deviations from the ATL values will be equal to those from the previous two cases. It is clear that, although the above three cases have widely varying pre-incident conditions, using the new features unifies the pre-incident conditions and focuses on the unmasked effects of the incidents.

The above example uses hypothetical illustrative occupancy-only numbers. Under actual operating conditions, the same logic illustrated in Table 6.5 and associated discussions still applies. Figure 6.5 shows the upstream and downstream occupancy values before and during two actual incidents from the SR-91 freeway in Orange County. In the Figure, points C1 and C2



**Figure 6.5 More Distinct Incident and Normal Patterns Using the Proposed Features**

represent incident conditions for incidents 1 and 2 respectively. Points N1 and N2 represent the normal pre-incident conditions. It can be seen in the figure that a vector such as O-C2 for instance is the sum of the two vectors; O-N2 which is the incident-free vector and N2-C2 which is the deviation-from-normal that the incident causes. The same applies for the other incident and the associated vectors. If the incident-free vectors are taken out (subtracted), and the deviations vectors are moved to start from the origin, the translated vectors O-C1' and O-C2' are obtained. The benefits are very obvious; [1] the incident-free points unify and collapse on the origin, [2] the incident points get closer to each other, and [3] the incident points become more distant and distinct from the normal pre-incident points, simplifying any subsequent classification attempts. The conclusion is that it is much simpler for a detection algorithm to operate on the deviations from the ATLS than to directly operate on observed values.

Simple shockwave analysis could be used to determine the time required for the shockwave to reach the detector locations. However, any modeling inaccuracies in the adopted shockwave analysis methodology (hydrodynamic theory for instance) will be automatically embedded in the incident detection model. To avoid such an error source, the shock wave arrival time could be easily determined from a time plot of the upstream and downstream occupancy values by observing the gap between the two values. The time at which the gap starts opening is the actual time of arrival of the shock waves

at the detector locations after the onset of the incident. The time at which the gap starts closing is also the time of arrival of the clearing shock wave after the removal of the incident. Such a way of marking traffic patterns for calibration or training purposes would mark the inputs as incident-related only when they start deviating from the normal class which in turn was previously “centralized” around the origin of the input feature space using the deviations from the ATLS as inputs.

### **6.4.3 Implementation of the New Features**

The objective in this section is to assess the potential transferability improvement from using the proposed features. The PNN was trained twice on the SR-91 simulation data, once using the conventional direct values of volume and occupancy and the second time using the deviations from the ATLS described above. The known volume and occupancy averages at each incident location, and during normal conditions (incident-free) were subtracted from the volume and occupancy inputs and only the deviations were presented to the network. Training vectors were marked as incident-related only after the arrival of the congestion shockwave to the detector locations. The delay before the arrival of the incident effects at the upstream and downstream loop detector stations depends on the speeds of the congestion shockwave traveling upstream and the clearing shockwave

traveling downstream as well as the incident location relative to the upstream and downstream stations. Since the effect of an incident does not necessarily reach both the upstream and downstream detector stations simultaneously, the incident effect arrival time was considered to be the time at which the upstream and downstream occupancies started to drift apart. This can be safely considered as an indication of the arrival of the incident-related shock waves either at the upstream and/or the downstream loop detector stations.

Similarly, the I-880 and I-35W databases were prepared. The average incident-free occupancy and volume for each incident were subtracted from the inputs and only the deviations were presented to the network.

After training the PNN on the SR-91 database, it was tested twice, once on the I-880 incident and incident-free databases, and another time on the I-35W incident and incident-free databases. The purpose of this testing was to assess the PNN's transferability without retraining using the new input features, and to compare the results to those previously obtained using conventional direct volume and occupancy inputs. In all cases, the DR and mean TTD were computed from testing on the incident data and the FAR was computed from testing on the incident-free data. Persistence tests of up to three intervals were used to further reduce the FAR due to random fluctuations of traffic.



Table 6.6 shows the testing results of the PNN on the I-880 incident and incident free data sets using the conventional and the ATL-based features. Figure 6.6 also shows the before and after performance envelopes. It is obvious from the results that training on the SR-91 freeway using the conventional features is not appropriate for the I-880 site, as was concluded earlier. The main reason is that the statistics of the two sites, using the conventional features, are significantly different, and consequently any direct transfer of trained algorithms between the two sites is not acceptable as the performance in terms of DR and FAR is significantly inferior to the TMC-acceptable values. Using the new features, however, tended to standardize the feature space and the related statistics. They standardize the pre-incident conditions, and the way an incident causes the traffic parameters to deviate from them. As the numbers indicate, the DR of the PNN was close to or higher than the 88% acceptable limit and the FAR was always lower than the 1.8% acceptable limit at all persistence intervals. Hence, using the proposed new features not only makes the direct transfer of the algorithms from one site to another feasible, but also the performance improvement over the conventional features was substantial. A relevant warning, however, is due in this case. The fact that the actual performance numbers were found to be within the acceptable limits should be interpreted with care, because this is data-dependent. It is reasonable to stress the importance of the relative evaluation in this case rather than the absolute

Persistence	conventional feature			ATL-based features		
	DR	FAR	TTD	DR	FAR	TTD
0	31	0.4	1080	93.3	0.01	181
1	29	0.02	1188	88.9	0	283
2	29	0	1218	86.7	0	226
3	27	0	902	86.7	0	256

**Table 6.6. Transferability to the I-880 Site Using Conventional and New Features.**

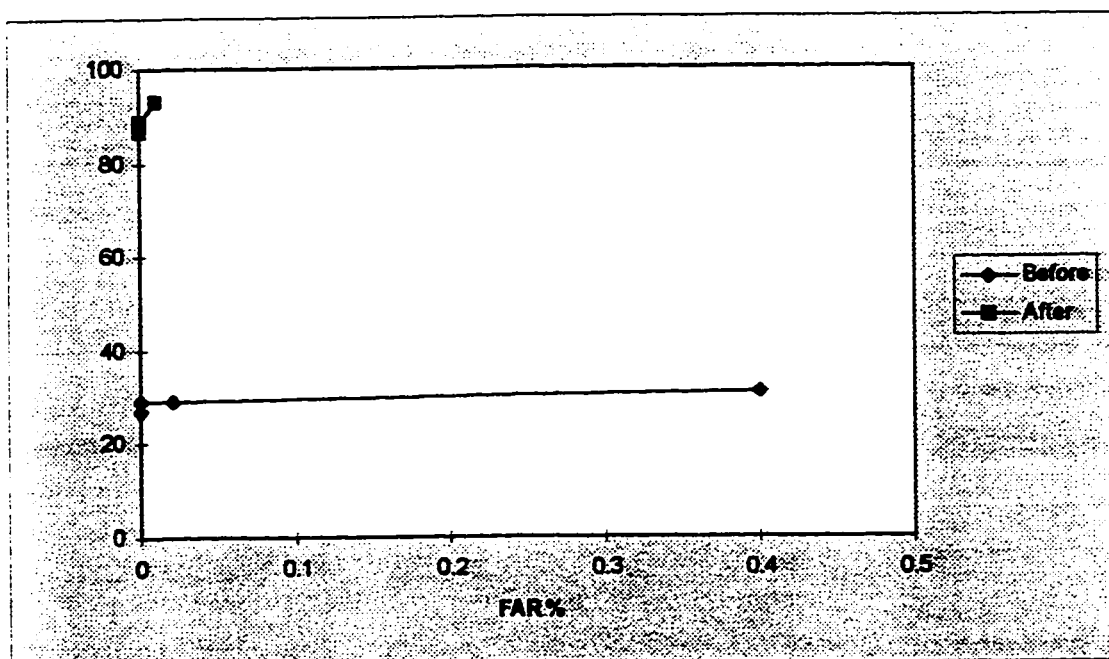


Figure 6.6. Performance Envelopes after Transferability to the I-880 Site Using Conventional and New Features.

one. Table 6.7 shows the corresponding results obtained from the I-35W site which fully support the parallel findings from the I-880 case. Figure 6.7 shows the before and after performance envelopes for the I-35W site.

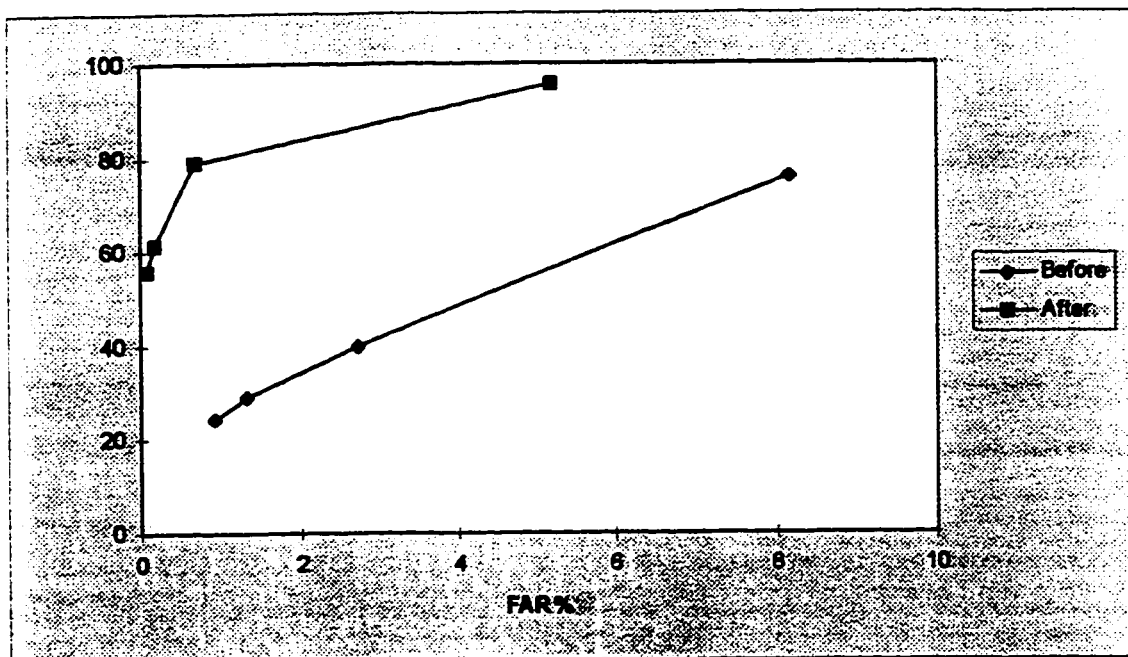
It is noticeable, however, that the improved performance is still far from ideal. Nevertheless, the gap between the improved performance using the ATL-based features and the desired ideal performance is much narrower than the corresponding case using the conventional features. This brings the role of training on-site in real-time back into the picture, the role of which becomes much easier with the improved transferability. As the PNN is statistically consistent and capable of improving in real time, its performance improves in service to further close this gap as was demonstrated earlier in this chapter.

## **6.5 POST-PROCESSOR INTERPRETER OF AID OUTPUT**

Loop-based incident detection algorithms rely on detecting the effects of an incident that may be reflected in the volume/occupancy-based input features. Due to the fact that these features are not perfect, i.e. they do not guarantee a distinct separation of incident patterns from incident-free patterns in the input feature space, as discussed earlier. The output of such algorithms will also be imperfect, which is not uncommon in the field of pattern recognition.

Persistence	conventional feature			ATL-based features		
	DR	FAR	TTD	DR	FAR	TTD
0	76.1	8.16	415	95.5	5.19	154
1	40.0	2.7	348	79.1	0.67	171
2	29.1	1.3	328	61.2	0.17	458
3	24.6	0.9	465	56.0	0.07	831

**Table 6.7. Transferability to the I-35W Site Using Conventional and New Features.**



**Figure 6.7. Performance Envelopes after Transferability to the I-35W Site Using Conventional and New Features.**

There is uncertainty involved in the binary decision of any incident detection algorithm as a generated alarm could be a true incident, or a false alarm. The current state of the practice in incident detection, which has been followed in this research up till this point, is such that the traffic variables, occupancy and volume are sampled once every certain time interval, most commonly 30 seconds apart. An input vector composed of a few minutes history of such data, both at the upstream and downstream loop stations is fed into the incident detection algorithm, either directly or after some preprocessing and transformations. The algorithm attempts to classify the input vector into one of two states, incident or incident-free. Once, the algorithm recognizes a single input vector to be incident-related, an incident alarm is issued at the Traffic Management Center (TMC) to initiate a verification/response procedure. This approach has several problems. First, the decision is based only on the latest 30 seconds input vector and is made in total isolation from previous decisions at the preceding intervals. Second, the generated alarm does not indicate the level of confidence in the presence of an actual incident in the field. Last but not least, these isolated decisions tend to be erratic and change from one interval to another very rapidly, alternating between incident and incident-free states. This rapid fluctuation could result in losing incidents that do not produce persisting successive incident-like vectors, and could also result in several false alarms in normal periods, as any erratic classification of a single input vector as an incident

would cause an alarm. For a reasonable detection rate, the number of false alarms and the associated waste of resources in the process of verification could outweigh the benefit from using the automated incident detection scheme. Consequently, the “crying wolf” algorithm may be abandoned by TMC personnel. It is not uncommon to use a persistence test of several intervals to cut down the false alarms. Although persistence tests increase the confidence in the incident decision, and reduce false alarms, they do that at the expense of missing weak incidents and delaying the detection of all others. In addition, in a large TMC jurisdiction, the absence of a severity or confidence measure tagging each alarm causes all alarms to be equal in importance, a problem that hinders the prioritization of responses and resource allocation to a set of alarms and potential incidents. Therefore, a simple sequential updating scheme is proposed for incident probabilities based on the well known Bayes’ theory and a graphical output that reflects the confidence in each incident alarm. This will be described in the rest of this chapter.

### **6.5.1 Drawbacks of the Isolated Binary Outputs**

Incident detection output usually takes a binary form of either “incident” or “incident free”, a process that gets repeated every, say, 30 seconds. No attempts have been made to translate these discrete binary outputs to a



continuous estimate of the probability of an incident in the field. To illustrate the conventional approach first, Table 6.8 shows two streams of incident detection output for two actual incidents from the SR-91 database. If a zero persistence interval is adopted, both incidents would be considered detected at time mark 150 seconds. However, if a persistence test of three intervals is used to cut down the number of false alarms, the first incident gets detected only at time mark 450 seconds, while the second incident goes undetected. Several drawbacks are associated with this approach which are identified as follows:

- Using a low persistence level may result in low confidence in incident alarms and a high false alarm rate during incident free periods due to short-term traffic perturbations
- Using a high persistence level may delay the detection of strong incidents and cause weaker incidents to go undetected.
- Using high persistence levels of  $n$ -intervals ignores all the information contained in previous successful classifications that lasted for less than  $n$ -intervals. For incident #1 in the above example for instance, and for a persistence test of three intervals, four successful classifications of the incoming traffic patterns were ignored prior to the time mark 360 seconds.

	Incident #1	Incident #2
Time	Output	Output
0	1	1
30	1	1
60	1	1
90	1	1
120	1	1
150	2	2
180	2	1
210	1	1
240	2	2
270	1	1
300	2	2
330	1	1
360	2	1
390	2	1
420	2	2
450	2	2
480	2	1
510	2	1
540	2	1
570	2	1
600	2	1
630	1	1
660	1	1
690	1	1
720	1	1
750	1	1

**Table 6.8. Sample Incident Detection Output for Two Incidents**

- There are no means of differentiating between incident alarms related to incidents of different strength. In the above example, and for the case of zero persistence interval, the two incidents were declared detected at time 150, and treated equally in terms of verification/response requirements regardless of the obvious weakness of incident #2. This weakness, however, would be obvious only if the entire stream of outputs is considered as a whole and not only the current output. Finally, if the number of incident alarms exceeds the verification/response capacity of a given TMC, a situation that is not uncommon, a prioritization scheme would be needed that assigns higher priorities to stronger incidents. This is obviously absent at present.

The alternative then is to attempt to link together the isolated 30-second decisions, combine the information contained in each one them and produce one estimate that gets updated sequentially every 30 seconds. This is achievable through the use of a Bayesian update process of a sequence of random tests. Each input vector is treated as a sample from the traffic population, the testing of which produces an incident or incident-free outcome. Knowing the prior probabilities, and as new samples are tested, the incident and incident-free probabilities get updated accordingly as described next.

### 6.5.2 Continuous Updating of Incident Probabilities

The process of continuous generation and updating of incident probabilities relies directly on the theory of total probability and Bayes' theorem (Ang 1975). As the given incident detection algorithm generates an incident alarm, the probability of existence of a true incident in the field could be computed using Bayes' theorem as follows:

$$P(I|A) = \frac{P(A|I) \cdot P(I)}{P(A|I) \cdot P(I) + P(A|F) \cdot P(F)}$$

where

- A : the event of an incident alarm generation by a given incident detection algorithm.
- I : the event of a true incident in the field.
- F: the event of an incident-free condition in the field.
- P(I): the prior probability of occurrence of an incident.
- P(F): the prior probability of an incident-free condition, which is the complement of P(I).
- P(A|I): the conditional probability of occurrence (generation) of an alarm given the occurrence of an incident in the field. This could be taken as equal to the correct classification rate of

incident related input vectors. It is algorithm-specific, and is obtainable during the testing of the calibrated algorithm.

$P(A|F)$ : the conditional probability of occurrence (generation) of an alarm given an incident-free condition in the field. This could be taken as equal to the incorrect classification rate of incident-free input vectors. It is also algorithm-specific, and is obtainable during the testing of the calibrated algorithm.

$P(I|A)$ : the conditional probability of existence of a true incident given the occurrence (generation) of an alarm.

Note that  $P(A|I)$  and  $P(A|F)$  are always less than unity because any incident detection algorithm is not perfectly reliable, and misclassifications inevitably occur.

Similarly, if the incident detection algorithm indicates an incident-free condition, the probability of existence of an incident in the field could be updated using the equation:

$$P(I|\bar{A}) = \frac{P(\bar{A}|I) \cdot P(I)}{P(\bar{A}|I) \cdot P(I) + P(\bar{A}|F) \cdot P(F)}$$

where  $\bar{A}$  is the complement of A, i.e. the event of the algorithm is indicating an incident-free condition at a particular time interval.  $P(\bar{A} | I)$  is also the complement of the  $P(A | I)$  as the conditioning component is the same.

The prior probability of an incident could be estimated on the basis of traffic, geometry, weather, and visibility conditions (see for instance, Madanat and Teng 1995). Once the prior probabilities are known, the output of the incident detection algorithm could be treated as a sequential test, every 30 seconds or so, that yields a true or false binary decision, i.e. incident alarm or incident-free indication. This output, together with the estimated prior probability of an incident, are plugged into the above equations in order to improve the prior information. In the same manner, the new calculated probability is treated as a prior probability for the next interval, together with the new output from the incident detection algorithm, to update the probability of an incident again, and the sequence goes on. The probability of existence of a true incident in the field is therefore generated and updated every 30 seconds. These probabilities could be presented to a TMC operator in a numerical format or as a rolling horizon graph that displays the current probability versus time.

In order to illustrate the above sequential updating process, the two incidents previously presented in Table 6.8 are revisited. The detection algorithm used in this example (the PNN) has a correct classification rate of incident vectors of 85%, and an incorrect classification rate of incident-free vectors of 4%. A starting prior probability of an incident of 5% is used in this example for the purpose of illustration only. i.e. the following inputs to equations 1 and 2 are used:

$$P(I) = 0.05$$

$$P(F) = 0.95$$

$$P(A|I) = 0.85$$

$$P(A|F) = 0.04$$

A minimum incident probability of 0.05 and a maximum of 0.99 are used to avoid “locking” the computations. Table 6.9 shows the updated probabilities for the two incidents given the outputs of the detection algorithm, i.e.  $P(I|A)$  or  $P(I|\bar{A})$ . Figures 6.8 and 6.9 illustrate the graphical display of these probabilities versus time for the two incidents respectively, and for the entire duration on the incidents.

It is very clear from Table 6.9 and Figures 6.8 & 6.9 that working with incident probabilities is more appropriate than just relying on an isolated decision based on one output at certain interval. Displaying the probability of an incident as a function of time gives a clearer picture of the evolution of the incident and severity of its effects on traffic. Also, it is very clear that incident #1 in the above example should be given verification/response priority relative to incident #2. A useful utilization of the resulting probabilities is to declare an incident only if the probability exceeds a certain limit (say 95%). In this case incident # 1 would be declared detected at time 180 seconds, and incident # 2 at time 450 seconds. This way, the rigid adoption of a certain persistence level and its associated problems could be completely avoided.

Time	Incident #1		Incident #2	
	Output	P(I)	Output	P(I)
0	1	0.05	1	0.05
30	1	0.05	1	0.05
60	1	0.05	1	0.05
90	1	0.05	1	0.05
120	1	0.05	1	0.05
150	2	0.53	2	0.53
180	2	0.96	1	0.15
210	1	0.79	1	0.05
240	2	0.99	2	0.53
270	1	0.94	1	0.15
300	2	0.99	2	0.79
330	1	0.94	1	0.37
360	2	0.99	1	0.08
390	2	0.99	1	0.05
420	2	0.99	2	0.53
450	2	0.99	2	0.96
480	2	0.99	1	0.79
510	2	0.99	1	0.37
540	2	0.99	1	0.08
570	2	0.99	1	0.05
600	2	0.99	1	0.05
630	1	0.94	1	0.05
660	1	0.70	1	0.05
690	1	0.27	1	0.05
720	1	0.05	1	0.05
750	1	0.05	1	0.05

**Table 6.9. Updated Incident Probabilities for the Two Incidents**



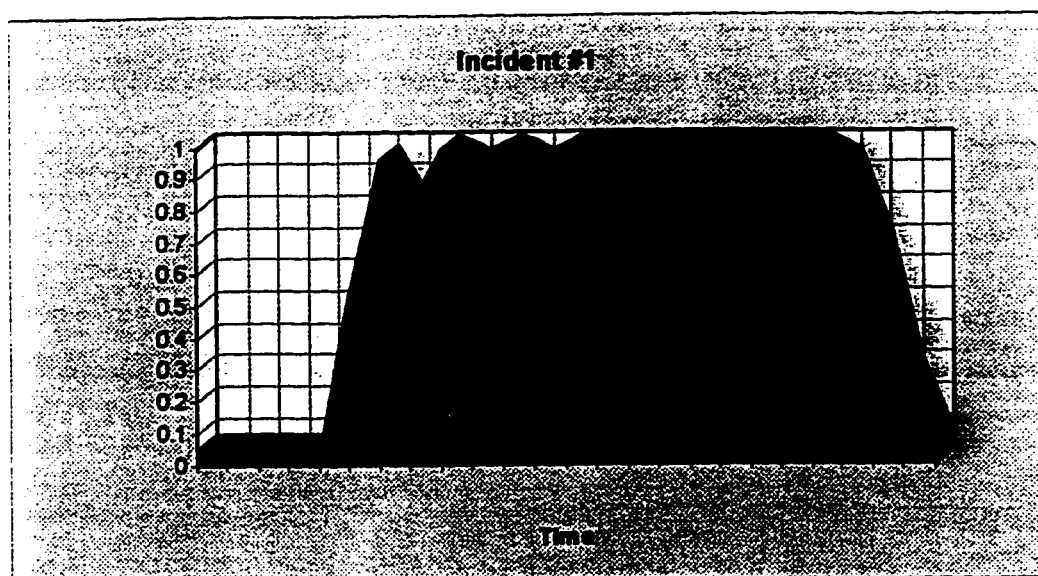
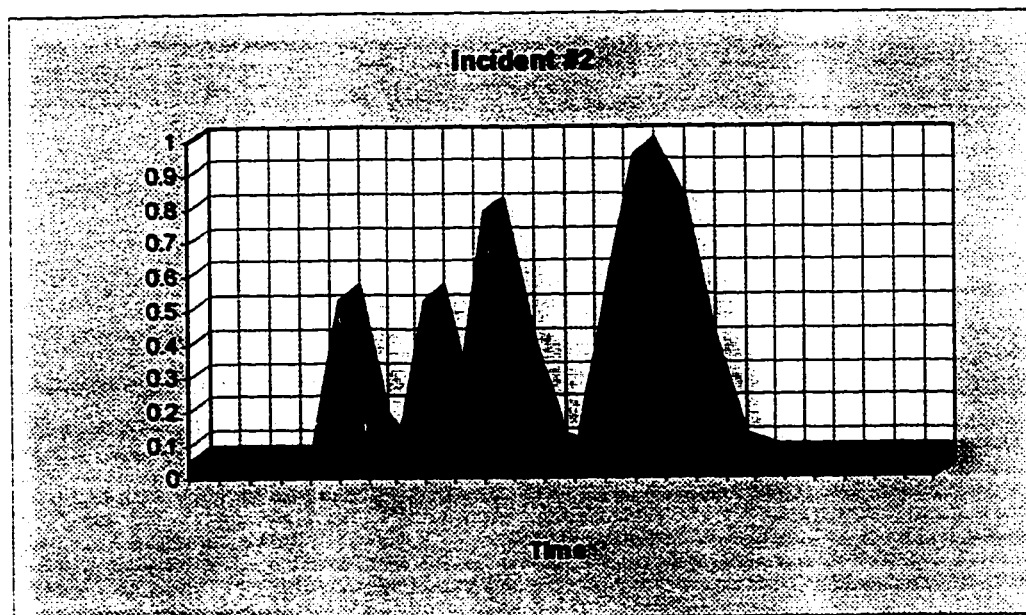


Figure 6.8. Updated Incident Probabilities vs. Time for Incident #1



**Figure 6.9. Updated Incident Probabilities vs. Time for Incident #2**

### 6.5.3 Alternative Updating Scheme of Incident Probabilities

The above scheme is applicable to the output of any incident detection algorithm including the PNN. However, there is another updating process that particularly fits only the PNN. The probability densities  $f_1(X)$  and  $f_2(X)$  produced by the PNN after classifying an input vector  $X$  can be used in the following equations to update the incident probabilities instead:

$$P(I|X) = \frac{f_1(X).P(I)}{f_1(X).P(I) + f_2(X).P(F)}$$

$$P(F|X) = \frac{f_2(X).P(F)}{f_1(X).P(I) + f_2(X).P(F)}$$

It is noticeable that the above equations are useable right after the probability densities are produced by the PNN and even before a classification of the incoming vector is made. Therefore there is no need to classify each incoming vector but rather update the posterior probabilities directly. Its application, however, is limited to the PNN, and since it is not functionally different from the general updating approach of using the binary outputs to update the posterior probabilities, the general approach is given preference. In cases where the PNN is the only algorithm under consideration, both approaches would be equally usable.

## 6.6 THE FULL UNIVERSAL SYSTEM

The components of the proposed universal freeway incident detection system are now complete. The PNN-based core is high in performance. The training process is very fast, actually instantaneous, and robust with no developer-dependent hand crafting of any complex parameters. Accounting for prior probabilities of incidents as well as the unequal costs of misclassification is an inherent component of the PNN theory. Using the new ATL-based features, the PNN transfers reasonably with non-catastrophic degradation in performance. The on-site learning capability of the PNN in real-time enhances the performance after transferability to fall well within TMC-acceptable limits. The posterior probability updating process mirrors the evolution of an incident in the field, reflecting its severity and captures the duration of its presence and time of termination. Also, the new input features make the framework immune to either real or virtual bottle necks or even loop biases. In fact, the overall framework fulfills the entire set of universality requirements as shown in Table 6.10 below.

The next chapter will explore a more complex version of the PNN presented in this chapter. A multi- $\sigma$  PNN will be developed and trained using a genetic algorithm. This only concerns the PNN-based core of the universal framework. The rest of the framework however remains the same.

Universality Attribute	PNN2
1. Highest performance.	√
2. Fast, robust and automated training and retraining	√
3. Reasonable TMC implementation requirements	√
4. Transferable logic:	√
5. Reasonably transferable training / calibration parameters	√
6. Minimal initial training data requirements	√
7. Account for prior probabilities of incidents	√
8. Account for the unequal costs of misclassifying traffic patterns	√
9. Capable of producing the posterior probability of an incident	√
10. Estimate incident severity	√
11. Capture incident duration	√
12. Statistical and theoretical soundness and clarity	√
13. Immunity to minor traffic fluctuations effects	√
14. Immunity to bottleneck effects	√
15. Immunity to consistent loop detector biases	√

x: absent attribute  
 √: fulfilled attribute  
 %: attribute fulfilled to some extent, but not fully.  
 N.A.: Not Applicable.

**Table 6.10. Universality of the PNN-based framework**

## CHAPTER 7

### AN IMPROVED GENETICALLY OPTIMIZED MULTI- $\sigma$ PNN

#### 7.1 INTRODUCTION

As discussed in Chapter 5, the original PNN uses a symmetric PDF estimation kernel. This symmetric kernel is the result of using a single smoothing parameter  $\sigma$  that applies equally to all the input dimensions, and using Euclidean distance as a measure of nearness of the different patterns. The use of such a symmetric kernel is inappropriate for many applications because it doesn't account for differences in variations along the axes nor the presence of correlation among the variables constituting the pattern vector.

A successful approach to this problem has been proposed, developed and evaluated earlier in this study and which relies on transforming the input variables into principal components. The major advantage of this approach is that the instantaneous training of the PNN is not jeopardized. Several other approaches to solving this problem however, have been proposed in the literature. Specht (1992, 1994, and 1996) used a different smoothing parameter for each input dimension. Traven (1991) and Streit (1990 and 1994) estimated the PDFs of the different classes as a sum of gaussian

kernels with a full covariance matrix. All these approaches have in common an iterative training procedure similar to that used by the MLF. Although performance improves, the training process is no longer instantaneous and is also prone to getting trapped in local minima as can the MLF. Very recent training procedures using genetic algorithms overcome the local minima problem and are faster than conventional training approaches. However, the training is still not instantaneous. A trade off remains between the gain in accuracy and the loss of the instantaneous training advantage. Training a multi- $\sigma$  PNN using a genetic algorithm is explored in this chapter.

## 7.2 A MULTI- $\sigma$ ADAPTIVE PNN

An important improvement to the PNN, called Adaptive PNN (APNN), is obtained by adapting a separate smoothing parameter for each input variable/feature dimension (Specht 1996, 1994, 1992). This often greatly improves generalization. It is known that each input contributes to the overall knowledge content of the network differently. Some poorly selected inputs even adversely affect the performance, or at least add complexity without benefit. Allowing a separate  $\sigma$  for each variable / dimension tailors these contributions on the basis of performance improvement.

The core of the single- $\sigma$  PNN is the estimation of a PDF as the sum of gaussian kernels which all have the simple covariance matrix  $\sigma^2\mathbf{I}$ , where  $\mathbf{I}$  is the identity matrix. Specht (1996, 1994, 1992) found that adapting a separate  $\sigma$  for each dimension greatly improves generalization accuracy. He used the two different adaptation methods described next.

### 7.2.1 Conventional Adaptation Methods

In the first method by Specht (1992), adaptation is accomplished by perturbing each  $\sigma$  a small amount and accepting the one perturbation which improves classification accuracy the most. Two criteria are used after each perturbation. First, the classification accuracy is determined using the PNN with the hold-one-out-method. One exemplar at a time is withheld from the training set, and the network is tested on this exemplar. The full evaluation is the sum over the entire training set. Second, the sum of probabilities is calculated using Bayes theorem. Using the hold-one-out-method once again, the probability of each held out pattern being classified into the correct category is equal to the PDF of the correct category evaluated for the held out pattern, divided by the sum of the PDF's for all categories. This measure provides a continuous measurement of classification accuracy so that improvements smaller than integer classification counts can be detected. The decision of which  $\sigma$  to change is made first on the basis of classification



accuracy. When more than one candidate yields the same classification accuracy, the sum of probabilities measure is used for the final selection.

In the second method (Specht 1994), adaptation is accomplished by perturbing each  $\sigma$  a small amount to find the derivative of the optimization criterion with respect to each  $\sigma$ . The conjugate gradient descent is used to find iteratively the set of  $\sigma$ 's that maximize the optimization criterion. The optimization criterion used emphasizes improvements in category separation only between categories where misclassification occurs. When patterns from category  $k$  are misclassified as members of category  $q$ , the likelihood ratios  $LR = f_k(X)/f_q(X)$  are calculated for all category  $k$  patterns, using the hold-one-out validation method. The mean log likelihood ratios are computed separately, and their ratio is taken. The ratio is summed over all cross categories where misclassifications have occurred. The following criterion is then minimized, which provides a continuous measurement of classification accuracy:

$$\sum_k \sum_{q \neq k} \frac{\text{Mean log LR for misclassified patterns (CAT}_k \text{ / CAT}_q \text{)}}{\text{Mean log LR for correctly classified patterns (CAT}_k \text{ / CAT}_q \text{)}}$$

The APNN not only finds a separate smoothing factor for each variable, but also detects variables that are poorly correlated with the desired output. All variables with a large associated  $\sigma$  have relatively small effects on the estimation of the PDFs. After the adaptation, the resulting  $\sigma$ 's can be used

to rank the input variable based on importance or remove some of them to reduce the dimensionality of the problem by getting rid of irrelevant inputs.

Although the above search mechanisms are more complicated than the simpler original PNN, the performance improvement might warrant such complexity. It is important to note that it is much easier to search the small  $\sigma$  space than to search the huge and complex weight space of the MLF. Besides, once the  $\sigma$ 's have been determined, the resulting network has the real time learning ability of the PNN. Nevertheless, if a more efficient and more robust mechanism for optimizing the choice of the smoothing parameters can be found, the APNN would benefit from the improved performance without the disadvantages of gradient descent search mechanisms such as difficulty of setting an optimal step size and getting stuck in local minima. The logical candidate alternative is to determine the optimal  $\sigma$ 's using a Genetic Algorithm (GA). This is motivated by the fastness, robustness, and efficiency of GA as a newly emerging optimization tool and the absence of the problems associated with gradient descent and similar search mechanisms. The Genetic Adaptive PNN (GAPNN) would be similar to the APNN above in the sense that it searches for a set of separate  $\sigma$ 's that optimize a performance criterion. The only difference however is that it uses a GA to search for the  $\sigma$ 's. The following sections elaborate on GA and the GAPNN.

## **7.3 GENETIC ALGORITHMS**

### **7.3.1 Introduction**

The field of Genetic Algorithms (GA) has been growing since the early 1970s, but only recently has it been increasingly applied to real-world problems. The field of GA is inspired by evolution in the real world, which is controlled by the process of natural selection. Organisms most suited for their environment tend to live long enough to reproduce, whereas less-suited organisms often die before producing children or produce fewer and/or weaker children with lower and lower chances of survival. Genetic algorithms can be defined as a problem-solving method that uses genetics as its model for problem solving, applying the rules of reproduction, gene crossover, and mutation to a population of candidate solutions or pseudo-organisms so those organisms can pass beneficial and survival enhancing traits to new generations (Chambers 1996).

In the real world, an organism's characteristics are encoded in its DNA. GA store the characteristics of artificial organisms in electronic chromosomes. A typical GA algorithm is linked to the real-world problem it is solving by an encoding mechanism and an evaluation function. The former is a way of encoding solutions to the problem on electronic chromosomes, most commonly in the form of bit strings. The evaluation function returns the worth or fitness of any chromosome in the context of the problem.

Given the initial components - a problem to be solved, an encoding procedure, and a fitness evaluation function - a GA can be used to carry out simulated evolution on a population of solutions as follows (Davis 1991):

1. Initialize a population of chromosomes (encoded candidate solutions to the problem).
2. Evaluate each chromosome in the population.
3. Create new chromosomes by mating current chromosomes; select two parents at a time, and apply mutation and crossover to create two new children. Most fit parents should have greater chance of being selected.
4. Delete members of the population to make room for the new chromosomes.
5. Evaluate the new chromosomes and insert them into the population.
6. Repeat till convergence is achieved or time is up.

Although the previous algorithmic description applies to most genetic algorithms, different researchers and practitioners implement this description in different ways. The most common or “typical” GA uses binary encoding, random initialization, roulette-wheel-parent-selection, and a one point cross-over and mutate operator. The following is a very brief overview of these terms. More details can be found in (Davis 1991).

Each solution to the problem is transformed into a binary string. At first the GA generates a population of initial solutions or chromosomes by filling each of these strings by randomly generated binary bits. The goodness of each of these solutions is evaluated and stored as a fitness record. This is the initial generation of chromosomes. The roulette-wheel-parent-selection (RWPS) mechanism is then applied to select two parent chromosomes for further mating and reproduction. The RWPS mechanism makes a list of available chromosomes, their fitness, and the running total of fitness, then generates a random number between zero and the accumulated total fitness, followed by selecting the first chromosome at which the running total of fitness is greater than or equal to the random number. This is analogous to the allocation of a pie-shaped slice of a roulette-wheel to each population member, with each slice being proportional to the member's fitness. Although, the effect of the RWPS mechanism is to return a randomly selected parent, each parent's chance of being selected is directly proportional to its fitness. On balance, over a number of generations, this will drive out the least fit members and contribute to the spread of the genetic material of the fittest population members. Once two parents have been selected this way, one-point crossover is applied. Crossover recombines the genetic material in two parent chromosomes to make two children. One-point crossover occurs when parts of two parent chromosomes are swapped after a randomly selected point. Extra diversity in the children is added by applying bit mutation. When bit

mutation is applied to a bit string, it sweeps down the list of bits, replacing each by a randomly selected bit if a probability test is passed. After a whole new generation of children is produced, they are evaluated and their fitness returned. Better solutions evolve as the process of reproduction and generational replacement proceeds. The process is terminated once it converges (does not improve further) or after a pre-specified number of generations.

### **7.3.2 Genetic Algorithms: General Uses and Advantages**

GAs are a very powerful new technology for searching for solutions to difficult problems. They have been applied to a range of function optimization, scheduling, and placement problems. From an engineering perspective, this new paradigm is a very powerful method for searching through a large and complex solution space featuring a large extent of local minima. In comparison to directed random search techniques, they are able to find optimal solutions quickly. In contrast to techniques like gradient descent, finding the optimum solution is not adversely affected by local minima. Therefore, it is more likely to find a global minima using a GA as opposed to gradient descent approaches. It is obvious then that the immunity to local minima traps and the speed advantages of GA make them a better candidate to train the APNN to optimize the search for the  $\sigma_s$ .

### 7.3.3 Genetic Synthesis and Optimization of Neural Networks

Both Genetic Algorithms and Neural Networks are biologically-inspired artificial-intelligence-based computational models. Both fields are maturing very rapidly, and are increasingly being utilized to solve real world problems. Since NN are known to be a very powerful pattern recognition tool, and GA are known to be very powerful optimization tool, the marriage of the two fields in order to optimize NN using GA is a logical step. After all, a large extent of biological neural networks are determined genetically.

Neural networks suffer from a lack of perspicuity as their behavior is not always explainable. Their massive parallelism, non linearity, and adaptive characteristics all conspire and make the analytic treatment of NN very difficult. The choice of network structure and parameters is an empirical-artistic exercise that relies on rules of thumb derived from past development experiences. The space of possible architectures and parameter combinations is extremely large. As a consequence, some significant amount of trial-and-error experimental hand-crafting is necessary before an adequate solution is achieved. It is impractical to rely on such guesstimation and trial and error to design networks for serious real-world problems. The empirical approach does not always produce a near-optimal network. Besides, a good solution might be data-dependent, and re-optimization might be necessary after every significant change in the application environment. The search for the best

attainable network structure and parameter-setting combination is a logical application for the genetic algorithm.

Genetic algorithms have been applied to the problem of NN design in several ways. For instance, Montana and Davis (1989) have explored the use of GA in training a NN of known structure. Belew et al (1990) used GA to set the learning and momentum rates for feed forward NN. Chang and Lippmann (1991) used GA to preprocess data in order to reduce the inputs to a NN without degrading performance. Harp and Samad (1991) explored using GA to discover the size, structure and parameters of a network to be trained by a separate NN learning algorithm. Koza and Rice (1992) looked at GA as a tool for developing architectures and weights together. More details on the subject of using GA for NN development can be found in Chambers (1996) and Winter et al (1995). Very recently, Ward systems (1996) used GA to optimize the smoothing parameters of a probabilistic NN which is very relevant to this research and will be explored further below.

#### **7.4 IMPROVED GAPNN-BASED INCIDENT DETECTION**

In this last phase of the research, the multi- $\sigma$  probabilistic neural network optimized by a GA as described above was used to develop a freeway incident detection algorithm. A shell implementation by Ward Systems (1996) was



used for the optimization process. The test sites and the data sets were the same as before. The following sections describe the performance of the GAPNN relative to the statistical distance PNN previously developed.

#### 7.4.1 Training and Testing of the GAPNN

The objective was to test the capabilities of the multi- $\sigma$  GAPNN relative to the PNN with a single smoothing parameter  $\sigma$ . To facilitate comparative analysis, training of the GAPNN followed the same training process of the PNN:

- First, the GAPNN was trained on the SR-91 database using the ATL-based input features, the same features used with the PNN. Both the GAPNN and the PNN were then tested on the I-880 database and the I-35W database to order to assess transferability of the GAPNN relative to the PNN.
- Second, the correctly classified incidents in the testing above were isolated in different databases. This was done on the basis of persistence test of three intervals in order to have high confidence in the detected incidents. These databases represented the traffic patterns that each network managed to detect, and hence can use to update its knowledge content and develop local-site-related statistics.

- These incidents were used to retrain each network. Both networks were retrained on the correctly detected incidents from the I-880 database, then tested on the entire I-880 database. The process was then repeated for the I-35W database. This tested the capability of each network to automatically update itself using incidents that it managed to detect in a totally new freeway environment, and to determine the extent of improvement. The statistical consistency of the PDF estimation techniques in the PNN family of networks, together with the nature of the associated training process, made this on-site retraining achievable in an unattended mode. Unlike other neural network paradigms such as the MLF, overtraining is not an expected problem, and hence once the new site-related data becomes available, the pattern updating process can proceed unattended.

Tables 7.1 and 7.2 show the relative performance of both the PNN and the GAPNN for the case of training on the SR-91 simulation data and testing on the real I-880 and I-35W databases respectively. Table 7.3 shows the two networks' performance on the I-880 data after retraining on the correctly detected portions of the same database. Table 7.4 shows their performance on the I-35W data after retraining on the correctly detected portions of the same database.

Persist	PNN			GAPNN		
	DR	FAR	TTD	DR	FAR	TTD
0	93.3	0.009	181	88.9	0.02	220
1	88.9	0	283	88.9	0	256
2	86.7	0	226	88.9	0	289
3	86.7	0	256	88.9	0	319

**Table 7.1 Performance of the PNN and GAPNN, trained on SR-91 data and tested on I-880 data.**

Persist	PNN			GAPNN		
	DR	FAR	TTD	DR	FAR	TTD
0	95.5	5.19	154	97.8	0.368	79
1	79.1	0.67	171	94.0	0.006	146
2	61.2	0.17	458	90.3	0.003	197
3	56.0	0.07	831	85.07	0.00	229

**Table 7.2 Performance of the PNN and GAPNN, trained on SR-91 data and tested on I-35W data.**

Persist	PNN			GAPNN		
	DR	FAR	TTD	DR	FAR	TTD
0	100	0.5	15	100	0	0
1	98	0	79	100	0	30
2	98	0	112	100	0	60
3	98	0	142	100	0	90

**Table 7.3 Performance of the PNN and GAPNN, trained on partial I-880 data and tested on I-880 data.**

	PNN			GAPNN		
Persist	DR	FAR	TTD	DR	FAR	TTD
0	98.51	0.011	48	100	0.12	15
1	98.51	0.009	85	100	0.05	48
2	98.51	0.006	116	100	0.02	78
3	97.76	0.003	147	100	0.01	111

**Table 7.4 Performance of the PNN and GAPNN, trained on partial I-35 data and tested on I-35 data.**

### 7.4.2 Comparative Evaluation and Concluding Remarks

Investigation of the above results yields the following several important observations and conclusions:

- Using the new ATL-based input features, the GAPNN yielded better results after transferability as compared the single- $\sigma$  PNN. This was particularly true for the I-35W database which contains non-lane-blocking incidents and in which a room for improvement exists. For the I-880 database, which contains strictly lane blocking incidents, both networks performed similarly and equally well.
- After on-site retraining, both networks approached perfect performance on both databases. This proves the importance of such a step after transferring an AID algorithm to a new site.
- Although the PNN did not perform as well as the GAPNN right after transferability to the I-35W site, it did perform as well after retraining.
- Being more accurate, the GAPNN detected more incidents than the PNN in a given time period after transferability. In other words, it detected the same number of incidents as the PNN in less time. Hence, it would require less time in service at a new site before an adequate performance could be reached.

- The better before-retraining performance of the GAPNN relative to the PNN comes at the expense of a longer training time. Although the genetic search mechanism is fast and efficient as opposed to other gradient-descent-like mechanisms, it is not instantaneous. This creates a trade off between the extra performance of the GAPNN on the one hand, and the instantaneous training capability of the PNN on the other. Since both networks achieve similar performance after retraining, the PNN might be more attractive for TMC-implementation.



## **CHAPTER 8**

### **SUMMARY, CONCLUSIONS AND RECOMMENDATIONS**

#### **8.1 SUMMARY**

In this research, a new potentially universal freeway incident detection framework has been proposed, developed and evaluated. The research effort was started by defining one possible comprehensive set of requirements that any universal incident detection algorithm or framework should fulfill. This set of universality requirements was used as a template against which all algorithms within the scope of this study were evaluated. Three major incident and loop detector databases were heavily utilized: one was an extensive simulated set of incidents using the INTRAS simulation model, the other two databases were unprecedented real databases collected from two major freeway sites in California and Minnesota, namely the Alameda County's I-880 freeway database and the Minneapolis' I-35W database. The universality of the most well known existing incident detection algorithms was tested using the above databases. Serious lack of universality and transferability in particular were detected in all existing algorithms. Prior to the development of the new universal framework, limits on acceptable performance were elicited from TMC surveys conducted as part of this effort.

Preliminary investigation of two promising advanced neural networks, namely the LOGICON and the PNN, was conducted. The PNN was more appealing due to its universality potential. The PNN was modified using a principal components transformation layer that resulted in performance enhancement, and lead to the choice of the modified PNN for in-depth development. The in-depth development stage was divided into three phases. The first was the extraction of an improved input feature set that produced more distinct classes in the input feature space. The new features enhanced transferability of the PNN and made the framework more compliant with the universality requirements. The second phase was the on-site real-time retraining of the PNN after transferability, a phase that produced near optimal detection performance. The third phase was the development of a post processor interpreter that linked the isolated 30 second outputs of the PNN and produced a sequentially updated probabilistic measure of existence of an incident in the field. The resulting overall PNN-based framework was found to be fully compliant with the entire set of universality requirements. Finally a new approach for training a multi-smoothing parameters version of the PNN was investigated. The approach utilized genetic algorithms for optimizing the selection of the multiple smoothing parameters. Obtained results indicated an improvement in performance over the single smoothing parameter PNN, but at the expense of longer training time.

## 8.2 CONCLUSIONS

The results of this research have demonstrated the superiority and universality of a particular neural network model, namely the PNN, for freeway incident detection. Adding a principal components transformation layer to the PNN was found to enhance its performance. Obtained results indicated the superiority of the PNN-based framework to other existing conventional and neural-network-based freeway incident detection algorithms. The components of the PNN-based framework were a preprocessor feature extractor, a PNN classifier and a post processor output interpreter, which were all found to be necessary for the overall framework to be universal. Although the genetically optimized version of the PNN showed better transferability, both versions showed equally good performance after retraining. The PNN was therefore concluded to be more practical for TMC implementation due to its instantaneous training capabilities.

## 8.3 RECOMMENDATIONS

Since the results obtained from the PNN-based universal framework were found to be very encouraging, the implementation of the framework in a real traffic management center environment seems to be the next logical step. The thrust of the framework was based on its capability to adapt to new field

conditions. Although this on-line promise was rigorously tested off-line in this research, it remains untested in a real TMC. The recommended installation of the framework within a TMC software platform would help assess the interaction of the TMC personnel with on-line updating process, and with the continuous probabilistic output of the system.

The developed system used volume and occupancy averages across all lanes as an input basis. No attempt was to use lane-specific data because of the potential limiting of the transferability of the resulting algorithm between sites with different number of lanes. This potential needs to be investigated further. As lane-averaged data from the upstream and downstream loop station can capture the temporal and spatial longitudinal patterns associated with an incident, lane specific data could make these patterns more vivid and add to them a spatial lateral pattern as well. This is due to the fact that, during an incident, vehicles tend to redistribute themselves across lanes and congestion decreases laterally as we move away from the affected lanes.

Finally, a new breed of loop-detector-based systems are now becoming available that can produce vehicle signatures and match signatures between the upstream and downstream stations. Valuable information could be extracted from this signature matching process that could enhance incident detection performance. Data collection efforts as well as a feature extraction process need to receive further attention using this new technology.

## REFERENCES

- Ang, A., and Tang, J. (1975), "Probability Concepts in Engineering Planning and Design.", Wiley and Sons, Vol. I: Basic Principles.
- 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. (1991), "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.
- Abdulhai, B., and Ritchie S. G. (1995), "Performance of Artificial Neural Networks for Incident Detection in ITS", ASCE Transportation Congress, pp. 227-238.

Baudry, M., and Davis, L., (1991) "Long Term Potentiation, A Debate of Current Issues", A Bradford Book, MIT Press.

Belew R., McInerney J., and Schraudolph N. (1990), "Evolving Networks: Using the Genetic Algorithms with Connectionist Learning.", CSE Technical Report CS90-174, Computer Science, UCSD.

Cacoullos, T. (1966), "Estimation of Multivariate Density", Annals of the Institute of Statistical Mathematics (Tokyo), 18(2), pp 179-189.

Chambers, L. ( 1996), " Practical Handbook of Genetic Algorithms.", Vol. I & II, CRC Press.

Chang, E.J., and Lippmann R. P. (1991), " Using Genetic Algorithms to Improve Pattern Classification Performance.", In R.P. Lippmann, J. E. Moody and D. S. Touretsky, "Advances in Neural Information Processing 3". pp797-803.

Cheu, R. L., and Ritchie, S. G. (1995), "Automated Detection of Lane-Blocking Freeway Incidents using Artificial Neural Networks", Transportation Research, Vol. 3C, No. 6, pp371-388.

Cheu, R. L. (1994), "Neural Network Models for Automated Detection of Lane-Blocking Incidents on Freeways.", Ph.D. Dissertation, University of California Irvine.

- Cheu, R. L., Recker, W. W. and Ritchie, S. G. (1993), "Calibration of INTRAS for Simulation of 30-Second Loop Detector Output", Preprint No. 930592, 72nd Transportation Research Board Meeting, Washington D.C.
- Cheu, R. L., Ritchie, S. G., Recker, W. W. and Bavarian, B. (1991), "Investigations of a Neural Network Model for Freeway Incident Detection", Artificial Intelligence and Civil and Structural Engineering, edited by B. H. V. Topping, Civil-Comp Press, pp. 267-274.
- Coultrip, R., Granger, R., and Lynch, G. (1992), "A Cortical Model of Winner-Take-All Competition Via Lateral Inhibition", Neural Networks, Vol. 5., pp. 47-54.
- Davis, L. (1991), "Handbook of Genetic Algorithms.", Van Nostrand Reinhold, New York.
- Forbes, G. J. (1992), "Identifying Incident Congestion", ITE Journal, Vol. 62, No. 6, pp. 17-22.
- Freeman, J. A. and Skapura, D. M. (1991), "Neural Networks: Algorithms, Applications, and Programming Techniques", Addison-Wesley.
- 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.

Giuliano, G. (1989), "Incident Characteristics, Frequency, and Duration on a High Volume Urban Freeway", *Transportation Research* Vol. 23A, No. 5, pp. 387-396.

Gregg, D. W.. and Nabrik, M. (1992), "The Projection Neural Network", *International Joint Conference on Neural Networks Vol II, IEEE*, pp 358-367.

Hall, A. L., Hall, F. L., Shi, Y., and Lyall, B. (1991), "A Catastrophe Theory Approach to Freeway Incident Detection", *Proceedings of the Second International Conference of Applications of Advanced Technologies in Transportation Engineering*, Edited by Stephanedes, Y. J. and Sinha, K. C., American Society of Civil Engineers, pp. 373-377.

Harp, S. A., and Samad, T. (1991), "Genetic Synthesis of Neural Network Architecture", in *Handbook of Genetic Algorithms*, Edited by L. Davis. pp. 202-221.

Hecht-Nielson, R., (1990), "Neurocomputing", Addison-Wesley.

James, S., and Chris, T. (1991), "Using Stereographic Projection as a Preprocessing Technique for Upstart", *International Joint Conference on Neural Networks Vol II, IEEE*, pp 441-446.



- Johnson, R. A., and Wichern, D. W. (1992), "Applied Multivariate Statistical analysis", Prentice Hall, Inc.
- Koza, J. R., and Rice, J. P. (1992), "Genetic Generation of Both the Weights and Architecture for a Neural Network.", International Joint Conference on Neural Networks, Seattle 92.
- 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.
- Madanat, S. and Teng, H. L. (1995), "A Sequential Hypothesis Testing Based Decision Making System for Freeway Incident Response", Final Report for the IDEA Program, Transportation Research Board, National Research Council, Washington D.C.
- Montana, D. J., and Davis, L. (1989), "Training Feedforward Neural Networks Using Genetic Algorithms.", Proceedings of the 11th international joint conference on artificial intelligence. pp. 762-767.
- NeuralWare (1993), "NeuralWorks Professional II/Plus" Neural Networks Software Manuals.
- Parzen, E. (1962), "On Estimation of Probability Function and Mode", Annals of Mathematical Statistics, 33, pp 1065-1076.

Payne, H. J., and Thompson, S. M. (1996), "Development and Testing of Incident Detection Algorithms: A Status Report on the Ongoing FHWA-Sponsored Project", Transportation Research Board 75th Annual meeting, Preprint #961358.

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.

Ritchie, S. G., Abdulhai, B., Parkany, E., Sheu, J., Cheu, R., and Khan, S. (1995), "A Comprehensive System for Incident Detection on Freeways and Arterials", proceedings of the Intelligent Transportation Society of America (ITS-AMERICA) conference.

Ritchie, S. G., and Cheu, R. L. (1993), "Neural Network Models for Automated Detection of Non-Recurring Congestion.", PATH Research Report, UCB-ITS-PRR-93-5, University of California Irvine.

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

Ritchie, S. G., and Stephanedes, Y. J. (1996), "Development, Testing, and Evaluation of Advanced Techniques for Freeway Incident

Detection.”, A Draft Interim Report, California PATH.

Rumelhart, D. E., McClelland, J. L., and the PDP Research Group (1986),

“Parallel Distributed Processing”, Vol. 1, MIT Press.

Simpson, P. K. (1990), “Artificial Neural Systems”, Pergamon Press.

Specht, D. F. (1996), “Probabilistic Neural Networks And General Regression

Neural Networks”, in Fuzzy Logic and Neural Network Handbook.

Edited by C.H. Chen, McGraw-Hill inc. pp. 3.1-3.44.

Specht, D. F., and Romsdahl, H. (1994), “Experience with Adaptive

Probabilistic Neural Networks and Adaptive General Regression

Neural Networks”, IEEE International Conference on Neural

Networks, Vol. II, pp. 1203-1208.

Specht, D. F. (1992), “Enhancement to Probabilistic Neural Networks”,

IEEE International joint Conference on Neural Networks, Vol. I, pp.

761-768.

Specht, D. F. (1990), “Probabilistic Neural Networks”, Neural Networks, Vol.

3, pp 109-118.

Stephanedes, Y. J. and Hourdakis J., (1996), “Transferability of Freeway

Incident Detection Algorithms”, Transportation Research Board 75th

Annual meeting, Preprint #961214.

- Stephanedes, Y. J. and Chassiakos, A. P. (1993), "Application of Filtering Techniques for Incident Detection", *Journal of Transportation Engineering*, ASCE, Vol. 119, No. 1, pp. 13-26.
- Streit, R. L., and Tod, E. L. (1994), "Maximum Likelihood Training of Probabilistic Neural Networks.", *IEEE Transactions on Neural Networks*, Vol. 5., pp764-783.
- Streit, R. L. (1990), "A Neural Network for Optimal Neyman-Pearson Classification.", *Proceedings of Joint International Conference on Neural Networks*, Vol. 1., pp685-690.
- Traven, H. G. (1991), "A Neural Network Approach to Statistical Pattern Classification by Semi-parametric Estimation of Probability density Functions.", *IEEE Transactions on Neural Networks*, Vol.2, pp 366-377.
- Ward Systems Group (1996), *NeuroShell2 manuals*.
- Lin, W. (1995), "Incident Detection with Data from Loop Surveillance Systems: The Role of Wave Analysis", Ph.D. Dissertation, UCB-ITS-DS-95-4, University of California Berkeley.
- Wicks, D. A., and Lieberman, E. B. (1980), "Development and Testing of INTRAS, a Microscopic Freeway Simulation Model", Vol. 1, Program

Design, Parameter Calibration and Freeway Dynamics Component Development, Report No. FHWA/RD-80/106, Federal Highway Administration.

Winter G., Periaux J., Galan M., and Cuesta P. (1995), "Genetic Algorithms in Engineering and Computer Science.", John Wiley & Sons Publishing.