Novel Event Classification for Structural Health Monitoring Systems

by

NISHANT J. DHRUVE

MASTER OF SCIENCE

2008

UNIVERSITY OF MANITOBA

Novel Event Classification for Structural Health Monitoring Systems

BY

NISHANT J. DHRUVE

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

MASTER OF SCIENCE

ABSTRACT

This thesis reports results obtained in applying neural networks to the problem of vehicle type classification from strain measurement data such as that obtained during structural health monitoring (SHM) of a vehicle bridge. It builds upon previous work which addressed the issue of reducing vast amounts of data collected during an SHM process by storing only those events regarded as being "interesting," thus decreasing the stored data to a manageable size. This capability is extended here by providing a means to group and classify these novel events using artificial neural network (ANN) techniques. In absence of actual strain measurements from a structure in service, simulated strain responses of cars, vans, buses and large trucks passing over sensors was generated and used for training and evaluation purposes.

Three types of neural systems consisting of a combination of supervised and unsupervised learning were investigated. The first consists of 2 layers of artificial neurons using both supervised and unsupervised learning. In this system, the first layer is a feature extraction layer and the second is the event classifier. The system was able to achieve an identification success rate of 63% for a dataset containing 3001 isolated vehicle strain patterns. The second system that is investigated is an extension of the first that included an extra data preprocessing stage. In this system, input data presented to the system is first scaled to the maximum value before being presented to the first layer. The scaling factor is retained and later presented to the second layer as an extra input. This system was able to achieve a success rate of about 92% for an isolated vehicle dataset containing 3001 data patterns. It was further found that proper identification of one vehicle when two are present within a single observation period was possible, even when the strain responses are overlapping. The vehicle type selected by this system in that case corresponds to the vehicle with the highest magnitude strain signature. Modifications to the

system were explored in efforts to improve recognition while removing the emphasis on magnitude alone. In doing so, a classifier was produced which selects the most consistently identified input pattern over a series of four sensors. This final system investigated is made up of sub-systems which consist of a data preprocessing stage and a two layer artificial neural network. Recognition accuracy for this system was found to be 85% for 3001 simulated vehicles. The system was found to do comparatively better than the neural classifier system with scaling for observation windows containing two vehicles.

ACKNOWLEDGEMENTS

I would like to first and foremost thank my advisor, Dean McNeill, for being very patient during the course of my Masters. He has been a great mentor, extremely helpful, and has always had an ear for everything.

In addition to that, I would also like to thank my family. They have done most and without whom I would not have been able to complete.

I would also like to thank my friends and colleagues who have been pushing me to achieve this goal. They have also made my time at the University of Manitoba quite enjoyable and memorable.

Furthermore, I would like to show my gratitude to the examining committee for the time and effort spent evaluating this work.

Lastly, I would like to thank my colleagues and managers at Elecsys Solutions and Norscan Instruments who have been understanding and patient with me during the course of my Masters.

Financial support for this work was provided by Intelligent Sensing for Innovative Structures (ISIS) and the Natural Sciences and Engineering Research Council of Canada (NSERC).

I. TABLE OF CONTENTS

II. LIST OF FIGURES

III. LIST OF TABLES

IV. LIST OF ACRONYMS

ANN	-	Artificial Neural Network
FSCL	-	Frequency Sensitive Competitive Learning
ISIS	-	Intelligent Sensing for Innovative Structures
NN	-	Neural Network
SHM	-	Structural Health Monitoring

1 INTRODUCTION

Civil structure technology has evolved over the years with engineers striving to make structures safer, efficient and long lasting. One of the ways that they have been able to achieve this objective is through the use of civionics. Civionics[1] is defined as the use of electronics in civil structures.

1.1 STRUCTURAL HEALTH MONITORING

Structural Health Monitoring (SHM) is the process of taking measurements such as strain, acceleration and temperature from civil structures in order to determine their "health." The monitoring process can be used by engineers to determine if a structure is safe and perform any repairs in order to extend its life or prevent potential catastrophes. Engineers, currently and in the past, monitored structures either through visual inspection, or using civionics, or a combination of both. They use civionics by load testing a structure such as a bridge, taking measurements and later analyzing the data to determine a structure's response to the load. SHM is slowly evolving into a continuous monitoring process where the structure is continuously monitored and analysed to assess its state of health.

SHM systems consist of sensors such as strain gauges, accelerometers and temperature sensors along with data acquisition systems being used collectively to take measurement data.

[1] Civionics was coined by Dr. Aftab A. Mufti of ISIS Canada at University of Manitoba, Winnipeg, Manitoba.

One of the most common structures where SHM systems are actively being used are bridges. Appendix A shows a sensor location diagram of the Red River North Perimeter Bridge in Winnipeg, Manitoba.

1.2 PROBLEMS WITH SHM SYSTEMS AND USE OF NOVEL EVENT DETECTION TECHNIQUES

Although active monitoring done on structures is seen as a very effective means of monitoring a structure's health, it has some problems associated with it. One of the major problems is the vast amounts of storage space that is required in order to store all the data.

If, for example, sensors having a sample frequency of 100Hz are placed on a bridge, a single sensor will, by definition, yield 100 data readings in a single second. If a bridge has 50 such sensors, this translates to 5000 data values every second, and 18,000,000 data values to be stored in just one hour. This shows that there is an incredible amount of storage space that is needed to store the measured data to be used later for processing. Engineers also have to face the daunting task of going through all that data in order to assess a monitored structure's behaviour.

Measured sensor readings of a continuously monitored structure usually represent the normal behaviour of the structure and are thus generally uninteresting. Therefore, one of the ways of reducing the amount of raw data is by storing only those chunks of data (events) that are interesting, otherwise known as novel events. Examples of such novel events could include seismic activity, abnormal forces on the structure, or even changes due to structural damage.

Previous work by Loren Card [5], addressed the issue of "large amounts of data collected by SHM systems" by providing a means to detect and store only those events that are considered interesting or "novel." The novel event detection is performed by using a combination of both signal processing and neural networks. He showed that it was effective and devised this detection technique on the Golden Boy located on top of the Manitoba Legislative Building, and the Taylor Bridge which are located in Winnipeg, Manitoba, and also the Portage Creek bridge which is located in Victoria, British Columbia.

1.3 ADDITIONAL DATA ANALYSIS

Large amounts of raw sensor data can therefore be reduced to a more manageable size through the use of artificial neural networks and signal processing. The reduced dataset could still be enormous and still present a daunting task of interpreting the data for the engineer. Part of this task could involve determining the cause of the novel event. The process of monitoring a structure could be further improved if classification of novel events were possible. The following research goes a step further by providing a means of identifying novel events and grouping them together to make any further data analysis easier. To accomplish this goal, the research focused on identifying vehicles traveling on a bridge by analyzing strain measurements with the use of artificial neural networks.

The main objective of this research is to identify vehicle type from a continuous stream of strain data without having to segment out individual vehicle responses for the purpose of classification. Secondarily we wish to explore the ability of a neural computing system to perform this classification task using data obtained from a sensor system already in place on a structure for other purposes. In addition, this work also looks at the robustness of the classification system to varying vehicle speeds, weights, and lengths.

There are several systems currently deployed that use different methods [7] to identify vehicles. Some of the identification systems use cameras to take pictures of the vehicles and then use high level image processing to determine the type of vehicle that passed through the sensor. Other commercial systems use multi-beam vertical light curtain or infrared detectors for identification and laser detectors to determine the speed of the vehicle and its position with respect to the sensor.

Another method that has been researched uses inductive loops and cameras to determine the type of vehicle. It was found in this research that different vehicles give different inductive signatures when they pass over these sensors. The sensor signatures have different shapes and properties. A neural network system using Kahonen maps was used in that research [7]. Another way of detecting vehicles is through the use of weight sensors. None of these identification systems seemed to be using strain measurements for identification of vehicles.

2 ARTIFICIAL NEURAL NETWORKS

The study of Artificial Neural Networks (ANN) was first inspired by the inner workings of animal brains. Brains contain networks of biological neurons that process inputs and are responsible for intelligent outputs such as muscle reflexes, interpretation of sight and even speech generation and interpretation. These biological networks are made up of very simple cells consisting of processing units called neurons and connection links between them called synapses.

A neuron [10] is made up of a cell body containing a nucleus with a single output connection known as an 'axon' and one or more input connections known as 'dendrites'. A dendrite is connected to another neuron's axon and the two together forms a communication link between the two neurons. Communication between dendrites and axons takes place through communication channels called synapses. Synapses are chemical junctions which act as channels for chemical signals. An impulse generated by a nucleus is converted to a chemical signal by the axon. The chemical signal is then picked up by the receptors at the dendrite and changed back to electrical signals. Neurons can have numerous input connections but only one output connection as indicated in figure 2-1.

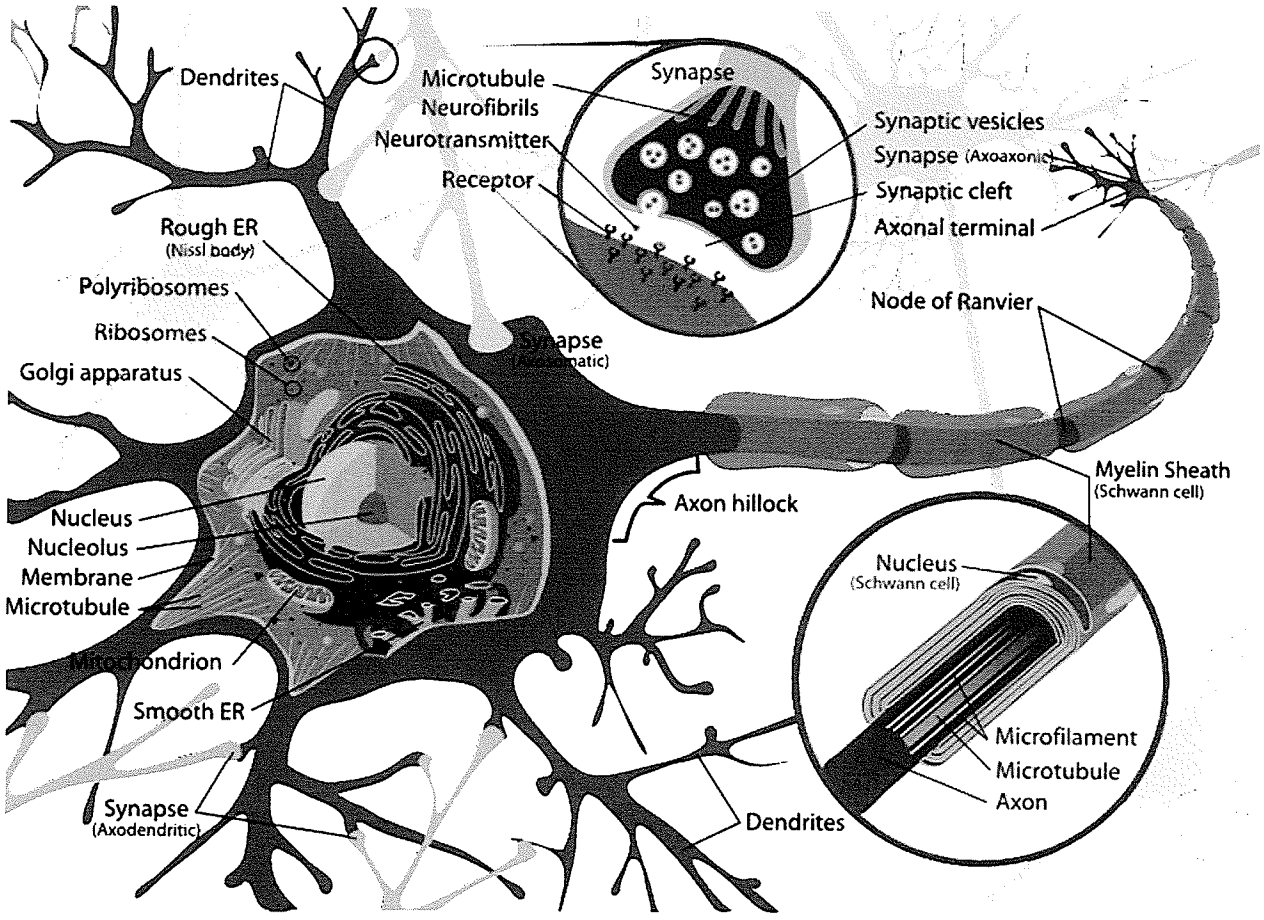

FIGURE 2-1: BIOLOGICAL NEURAL NETWORK [8]

The neurons, also known as the processing units, take inputs from the other connected neurons or sensor nerve endings and process them. This biological process is a weighted sum of all the inputs to the neuron. An output is generated by the processing neuron if the input weighted sum is greater than a threshold value. No output signal is generated if the weighted sum is less than the threshold. The output is then passed onto other neurons for further processing. A biological network is an interconnected mesh of numerous neurons that process inputs and perform simple and complex tasks.

The brain is able to process information at a very high rate even if a single neuron processes information at very slow speeds. A brain's high speed processing is possible through parallel

computations taking place among the neurons, which number approximately 100 million with the synapses/communication channels numbering approximately 100 trillion [12].

Artificial Neural Networks (ANNs), which are usually programmed into computers, are modeled after biological neural networks. These networks are comprised of neurons and weighted communication links; neurons are modeled after biological neurons while the communication links are modelled after dendrites, axons and synapses present in biological neural networks.

ANNs have emerged as a very valuable technology in many applications that deal with pattern recognition. They are employed in situations where a deterministic algorithm is not available to solve the problem and also where the relationship between inputs and outputs is very vague. Some of the applications where neural networks are used include character recognition, speech recognition, remote sensing, geophysical prospecting, medical applications, and many image processing applications.

Artificial neural networks are part of a bigger system that is used in previously mentioned applications. Such a system pre-processes the data before it is presented to the network. The network then takes the presented pre-processed data and further processes it to determine the likely output for that data. The outputs can then be processed again before being given to the end user. The structure of such a system is given in figure 2-2.

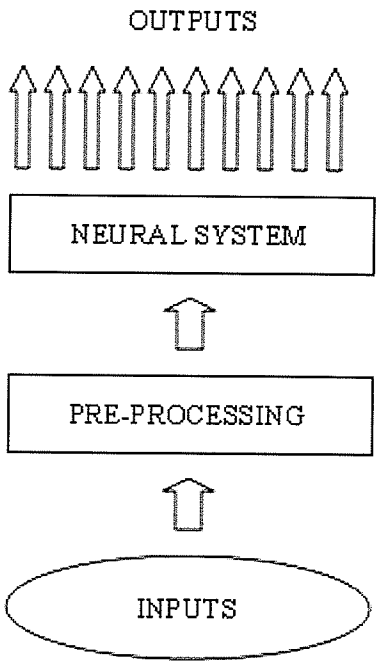

FIGURE 2-2: NEURAL NETWORK SYSTEM

Figure 2-3 depicts the basic structure of a two layer artificial neural network where the neurons are connected to inputs and to outputs. Just as in biological networks, a neuron in an ANN can have multiple inputs but only one output. These inputs, outputs, and the neurons are connected to each other through weighted communication links. Each neuron takes a weighted sum of the inputs and passes the sum through a monotonically increasing non-linear function known as an activation function to generate an output or activation. The output is then either passed to other neurons or is used as a decision of the network.

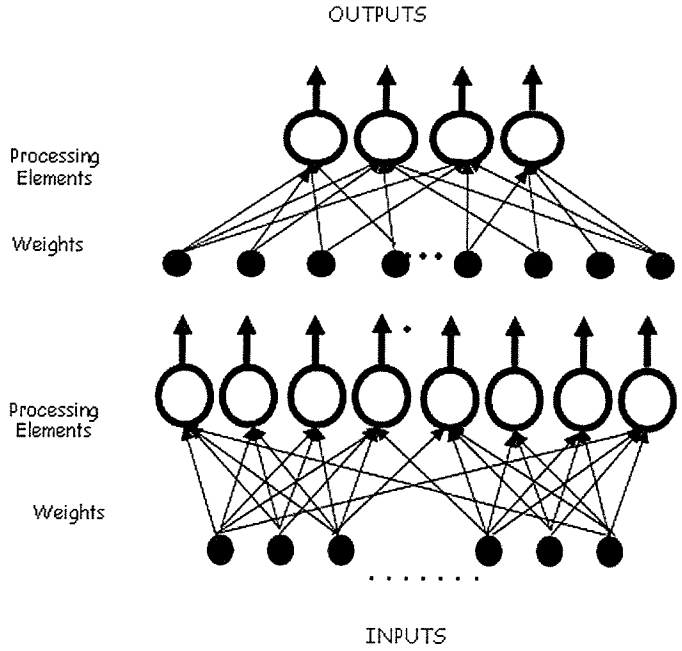

FIGURE 2-3: NEURAL NETWORK STRUCTURE

The processing speeds of ANNs are dictated by the speeds of a processor and amount of parallelism that can be achieved during processing. Their processing speeds are much slower than that of a brain because of a constraint on the number of nodes that can be deployed in an ANN. Higher processing speeds can be achieved by using faster and/or parallel processors.

There are many different kinds of neural network architectures and learning algorithms, and their use depends on the type of application. Networks are first trained before being deployed in their respective applications. Training of a network usually involves modification of weights associated with the communication links to give a desired output. Numerous different methods exist in the way these networks are trained but these can be divided into two major categories; supervised learning and unsupervised learning. In supervised learning algorithms the ANN is presented with inputs and known outputs during the training process. On the other hand, in unsupervised learning the training network is presented with just the inputs during the training process. The following project uses both these types of learning algorithms for its ANN system.

2.1 UNSUPERVISED LEARNING

As mentioned previously, the training process during unsupervised learning consists of presenting the network with just the data inputs. The network learns the pattern of this input data during training. Once the training is completed, the network will behave in such a way that the neuron whose weight vector most closely matches any new input data will give the highest or strongest activation.

Unsupervised learning may make use of competitive learning algorithms to train the weight vectors associated with each neuron. These algorithms consist of competitions among neurons, based on their ability to represent the input. The variables used for the competitions depend on the type of learning algorithm that is used. The competitions are usually followed by modification of neuron weight vectors thus enhancing a neurons ability to represent any new inputs.

There are several variations of unsupervised competitive learning algorithms some of them include *hard competitive learning,* [17,18], *soft competitive learning* [17,19] and *frequency sensitive competitive learning.* This project makes use of the frequency sensitive competitive learning algorithm to train its unsupervised artificial neural networks. This learning algorithm was used because of its straightforward implementation and its ability to better group input data when compared to the other competitive learning algorithms.

2.1.1 FREQUENCY SENSITIVE COMPETITIVE LEARNING

The variant of *Frequency Sensitive Competitive Learning (FSCL)* that is used in this research was first proposed by Krishnamurthy et al. [1,2,3] in the 1990's. As with all competitive learning algorithms, FSCL makes use of a cost function for competitions among neurons during the training process. The cost function in the case of this variant of FSCL is comprised of a fairness function and the Euclidian distance between the weight vector and the inputs. The cost of representing an input $\boldsymbol{x}(n)$, is represented by h_i for a neuron i as shown in equation 2-1. The n in this equation represents the n^{th} input.

$$h_i(n) = F(u_i(n)) \times \|w_i(n) - x(n)\| \quad (2\text{-}1)$$

During the competitions, the node (neuron) that results in the lowest cost h_i for a particular input is regarded as the winner of that competition. The weights of the winning node are then changed.

The fairness function, $F(u_i(n))$, is chosen as a "non-decreasing" [3, 5] function of the number of competitions a node i wins. The number of competitions a neuron wins is denoted here by u_i. The function is typically chosen as a linear function of u_i. For this research the fairness function was chosen as the number of competitions a node wins as shown in equation 2-2.

$$F(u_i) = u_i \quad (2\text{-}2)$$

The main purpose of the fairness function is to limit the number of wins of one particular node. This mechanism ensures that a single node does not keep on winning all the subsequent competitions but rather gives a chance to other nodes in the vicinity of the data. If a node wins numerous competitions its count increases and so does the fairness function, which eventually translates to higher cost. Therefore a node with a large number of wins has a lower chance of winning competitions due to higher cost and this gives a chance to other nodes to win the competitions. Therefore the Fairness function, $F(u_i)$, introduces a "count-dependant" weighting to the cost functions during competitions.

The advantage of using a fairness function in cost calculations is that it averages the number of competitions among the neurons and therefore achieves an almost equal weighted representation of data among nodes. This helps in that it assigns each node a data cluster thus preventing the problem of isolated nodes.

The problem of isolated nodes or data clusters is seen in some other types of competitive learning algorithms [16,20]. An example of this behaviour is illustrated in figure 2-4. The plot on the left in figure 2-4, which is a result typical of hard competitive learning algorithm, shows two nodes A and B representing a small cluster of data points while two other smaller clusters are represented by only a single node C. This plot shows an unbalanced data representation among the neurons. This would have been prevented if FSCL were used. As can be seen in the plot on the right of the same figure each data cluster is represented by single nodes.

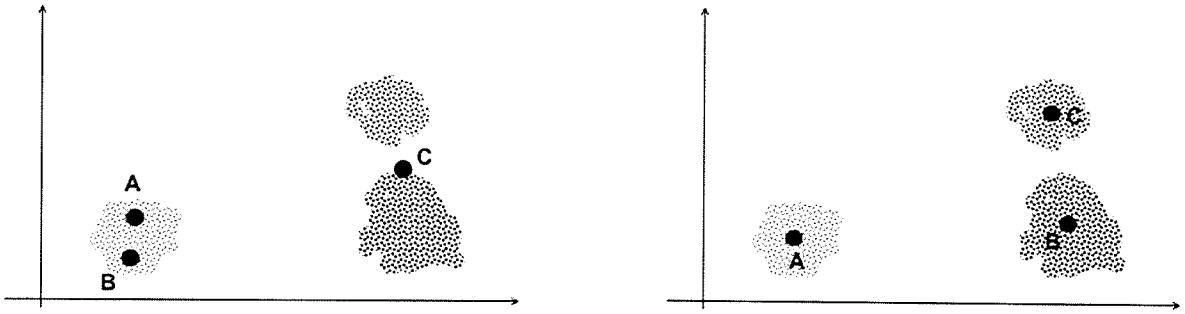

FIGURE 2-4 COMPARISIONS BETWEEN COMPETITIVE LEARNING AND FSCL

During training, the weight vector of the winning node, i^*, is changed by a small fraction of the input vector. The factor is known as the learning rate factor which ensures that the weight vector of the winning node moves closer to the input. Equation 2-3 represents the equation used to calculate the new weight vector, $\boldsymbol{w}_{i^*}^{new}$, for the winner node i^*.

$$\boldsymbol{w}_{i^*}^{new}(n) = \boldsymbol{w}_{i^*}^{old}(n) - \varepsilon\left(\boldsymbol{w}_{i^*}^{old}(n) - \boldsymbol{x}(n)\right) \tag{2-3}$$

The learning rate factor is chosen to be between 0 and 1. It should be kept small "in order to ensure that the overall statistics of the data distribution are used" [11]. A large factor would result in a single data vector having a larger impact on the weight vector. For this thesis, the value of ε = 0.001 was chosen as a compromise between the conflicting goals of reducing learning times and representing the overall statistics of the data.

2.1.2 FSCL TRAINING ALGORITHM

The training starts off by initializing each neuron's weights and their associated number of wins. The weights of a neuron can be initialized in a number of different ways; they can be initialized to either random values, constants, or even the first few data input vectors. The initialization vector usually, impacts how long it will take to train the network and also what data cluster each neuron represents. The frequency or number of wins of each neuron is initialized to 1 so that the cost for the first iteration is not equal to zero.

The training then begins by introducing an input vector to each node in the network. Each node processes the inputs to calculate the cost incurred if it is to represent that particular input vector. The node having the least cost is then labelled as the winner of that competition and its weight vector is modified to reflect the input vector it now represents. The frequency of the winner node is incremented. The training then continues with introduction of another input vector, calculation of the cost, and finally updating the weights and frequency of the winning node.

An error is calculated at each iteration of the learning process to determine when to stop training the network. The learning process is deemed completed when the average error over the training set converges, or in other words the rate of change of the average error becomes negligible over an extended period of time. The following research uses mean squared error, E, between the winning node's weight vector and the input data vector for average error calculations. Equation 2-4 represents the mean squared error function.

$$E = \frac{1}{N}\sum_{n=1}^{N}\left(\boldsymbol{w}_{i^*}(n) - \boldsymbol{x}(n)\right)^2 \qquad (2\text{-}4)$$

In equation 2-4, $\boldsymbol{w}_{i^*}(n)$ is the weight vector of the winning node during the n^{th} input and N is the total number of data vectors in the training dataset.

Note 2-1 gives a brief description of the FSCL training process.

FSCL learning algorithm

i. *The number of wins of all the nodes is initialized to 1.*

ii. *Initialize weight vectors of neurons to either constants, random numbers or input vectors.*

iii. *Introduce a new data input vector to the network.*

iv. *Calculate the Euclidian distance between the input vector and the weight vector of each neuron*

v. *Calculate the cost using the function given in equation* 2-1

vi. *The neuron with the lowest cost is the winner neuron. Increment its number of wins.*

vii. *Modify the weights of the winning neuron using equation* 2-3.

viii. *Calculate the squared error of the current learning iteration*

ix. *If end of the dataset has not been reached, then go to iii.*

x. *End of dataset has been reached so calculate the average error using equation* 2-4

xi. *Go to iii, if the rate of change of error over an extended period of time has not become negligible.*

xii. *Done training.*

NOTE 2-1: THE FREQUENCY SENSITIVE COMPETITIVE LEARNING ALGORITHM

2.2 SUPERVISED LEARNING

As mentioned earlier, in supervised learning a network is presented with input data vectors as well as known outputs. The network has to be trained such that it forms a relationship between the inputs and the known outputs by training its weight vectors. Supervised learning is used when there is prior knowledge of the desired output behaviour of a network to particular inputs.

The training algorithm is presented with an input dataset along with the desired output behaviour. In the case where only a single neuron is to represent the desired output, the desired output activation of that neuron will be non-zero while the rest of the neuron activations are zero. For the following research the supervised learning algorithm is provided with the input data and an output that identifies the neuron that is to represent that particular input.

The main purpose of providing the identity of the "winner" neuron is to guide the learning such that the network learns to encode the desired input-output relationship with the intent that the output will be the same if a new similar input vector is presented to the network. Only the weight vector of the neuron that was picked to represent that input is modified according to the input data that they are to represent.

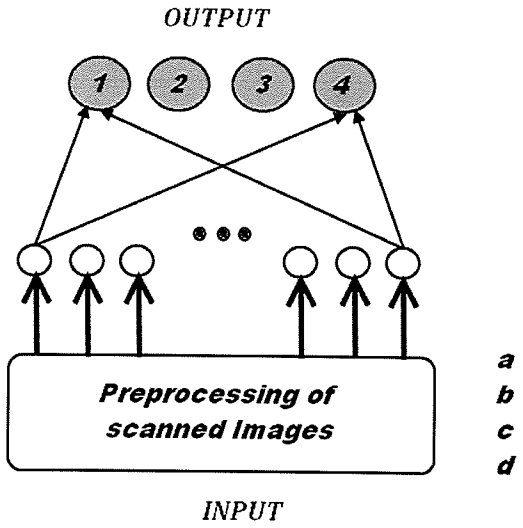

FIGURE 2-5: CHARACTER RECOGNITION NETWORK

For example, the network shown in figure 2-5 is a hypothetical character recognising network for the characters a, b, c, ands d represented by output neurons 1, 2, 3, and 4 respectively. The input is a scanned image of a hand written character. During training of the final neural layer, the supervised learning algorithm is presented with prior knowledge identifying the training input data and the target output. For example if the current training input data is the handwritten character 'a', the training algorithm will be given this input and told that neuron 1 is to win the competition.

The weight vector of the chosen node i^* is modified as shown in equation 2-5. The ε is the *learning rate factor*, $\boldsymbol{x}(n)$ is the n^{th} input vector while $\boldsymbol{w}_i(n)$ is the weight vector.

$$\boldsymbol{w}_{i^*}^{new}(n) = \boldsymbol{w}_{i^*}^{old}(n) - \varepsilon\left(\boldsymbol{w}_{i^*}^{old}(n) - \boldsymbol{x}(n)\right) \tag{2-5}$$

The new weight vector is denoted by $\boldsymbol{w}_{i^*}^{new}(n)$. The learning factor just as in unsupervised learning was chosen as 0.001 which has been found suitable during training. The learning factor is kept small to ensure that the weight is not based solely on new inputs but a portion of the whole input dataset. This is why the node weights are changed by a small portion of the input vectors.

Supervised learning also utilizes an error measure to keep track of the training that is taking place. The main purpose of this is to ensure that the network is not under trained or over trained. The mean square error is used as the error measure for this project. The error E is illustrated in equation 2-6, $\boldsymbol{x}(n)$ is the current data input vector while $\boldsymbol{w}_{i^*}(n)$ is the weight vector of the winning node during the n^{th} input.

$$E = \frac{1}{N}\sum_{n=1}^{N}\left(\boldsymbol{w}_{i^*}(n) - \boldsymbol{x}(n)\right)^2 \tag{2-6}$$

Once the training is completed, new data can be presented to the network for classification. The outputs from the nodes is the cost of representing that input data. This project utilizes Euclidian distance as a cost measure during competitions. In equation 2-7, $\boldsymbol{x}(n)$ is the n^{th} input data vector, and $\boldsymbol{w}_i$ is the weight vector of node i.

$$y(n) = \left\|\boldsymbol{w}_i - \boldsymbol{x}(n)\right\| \tag{2-7}$$

Note 2-2 gives a brief description of the algorithm used to train the network.

Supervised Learning Algorithm

i. *Initialize weighs vectors of all neurons to either constants, random values or input vectors.*

ii. *Introduce a new input data vector to the network.*

iii. *In the case of this thesis the output behaviour of the network is the desired winning neuron.*

iv. *The weight vector of the desired winning neuron is changed using equation 2-5.*

v. *Calculate the squared error between the input data vector and the winning neuron weight.*

vi. *Go to (ii) if end of the training dataset has not been reached.*

vii. *Calculate the average mean squared error using equation 2-6.*

viii. *Done training if error converges otherwise go back to (ii).*

NOTE 2-2: SUPERVISED LEARNING

3 DATA MODELS

The results obtained in this research were acquired using simulated SHM data from a simplified bridge model that was based upon the characteristics of the Red River North Perimeter Bridge located in Winnipeg, Manitoba. This particular bridge was chosen because it is a high traffic, GFRP steel free deck bridge and was recently fitted with 48 gauges to monitor its behaviour. It consists of 2 eastbound and 2 westbound lanes, and is made up of 10 spans. Each span has a split eastbound and westbound deck that is supported (nominally) by 4 girders.

A wide variety of traffic can be seen traveling on this bridge, consisting of small to medium sized cars and also light to heavy weight large trucks. Figure 3-1 is a picture of this bridge.

FIGURE 3-1: PICTURE OF THE RED RIVER NORTH PERIMETER BRIDGE

Most of the 48 gauges that are fitted to the bridge are of the metal-foil type with some being fibre gauges. The data that has recently been collected has been from the metal foil sensors. Figure 3-2 illustrates some real strain bridge data obtained from the sensors located on a girder. The data was collected from gauge ESGgirder1 during a controlled load test of 2 trucks travelling at approximately 100km/hr in the passing lane. This particular data was collected on 2007-07-12 at 10:31:15. It can be seen from these graphs that the strain signatures of both trucks have almost 2 significant peaks.

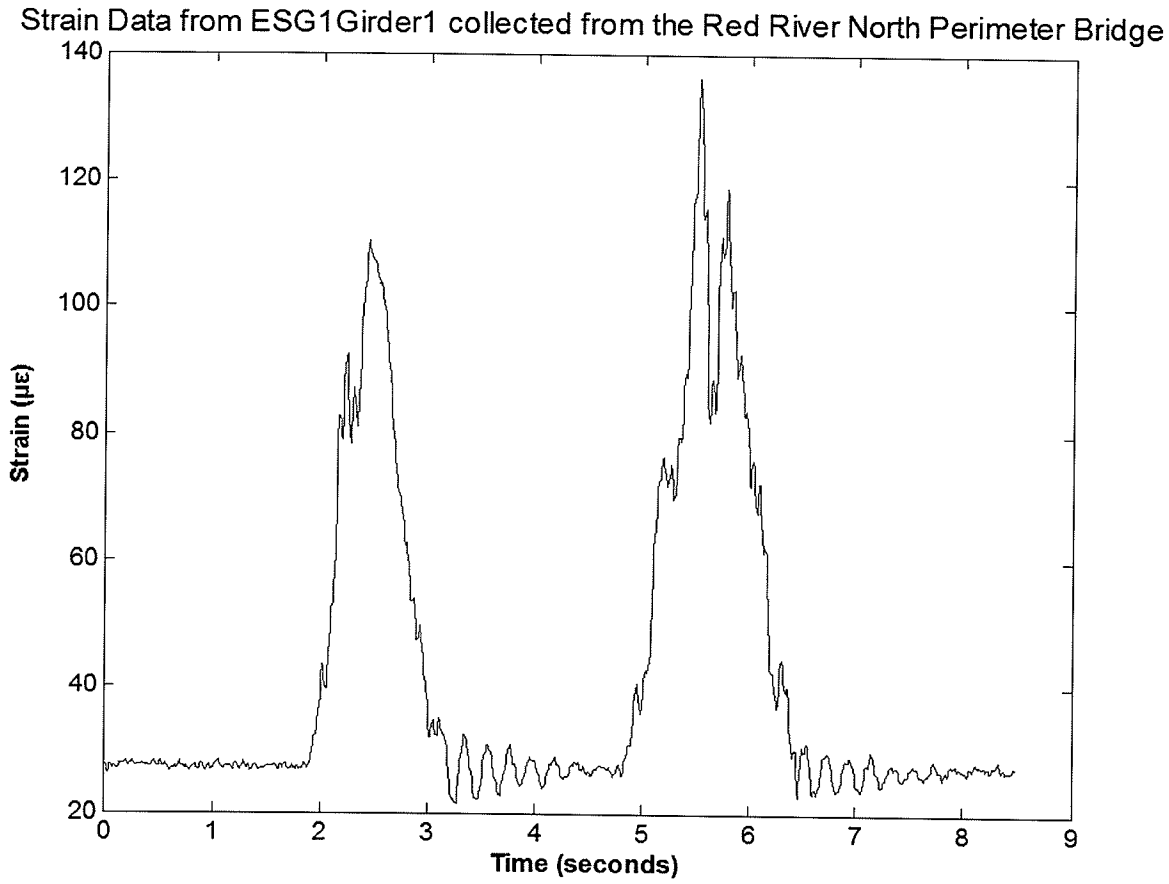

FIGURE 3-2: STRIAN DATA COLLECTED FROM THE RED RIVER NORTH PERIMETER BRIDGE

Figure 3-3 illustrates the installation of sensors. A data acquisition system is added at the receiving end of the sensors to collect the sensor readings.

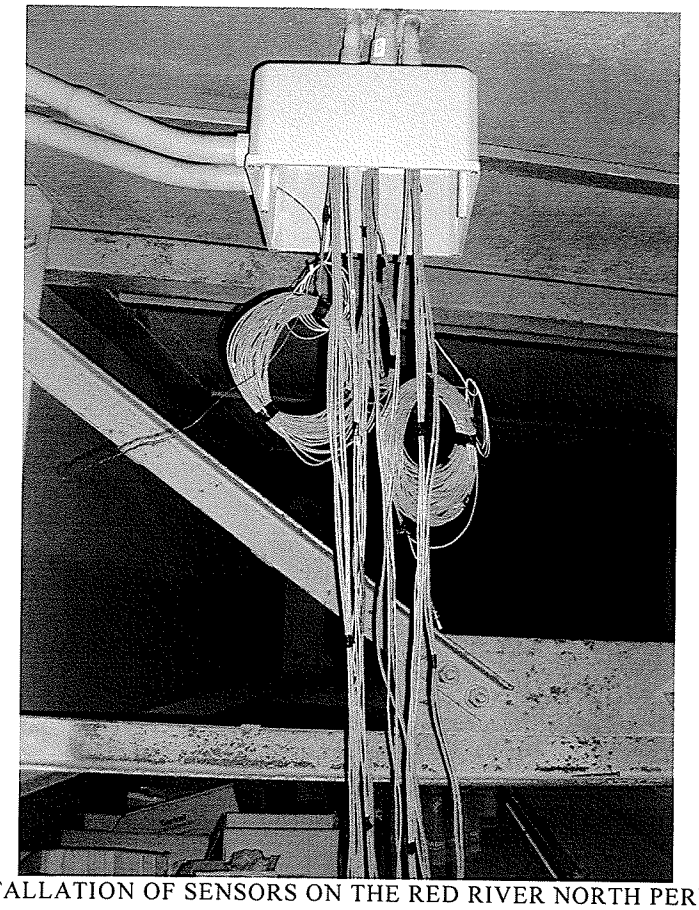

FIGURE 3-3: INSTALLATION OF SENSORS ON THE RED RIVER NORTH PERIMETER BRIDGE

3.1 COMPUTER MODEL OF THE NORTH PERIMETER BRIDGE

Due to the lack of actual measurement data from the bridge during the course of this research, a highly simplified computer model of the bridge sensors was created in order to generate test strain data which could be used for training, testing and validation of the *Neural Network Systems*. The model, illustrated in figure 3-4, was created in software, as a single deck of length 12.67 meters corresponding to the length of the eastbound approach part of the Red River North Perimeter Bridge. A subset of 4 simulated gauges of the 48 gauges were placed four meters apart on the span. Simulation of the bridge was done using MATLAB® from MathWorks [4] because of its relative ease of use and data computing capabilities.

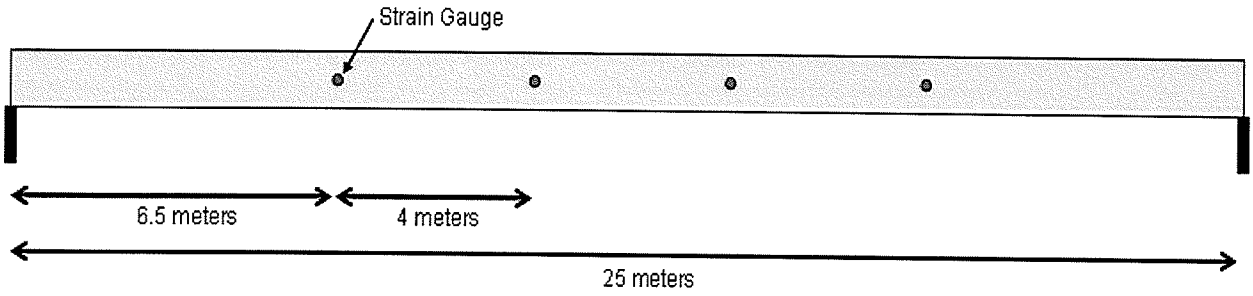

FIGURE 3-4: LOCATION OF SENSORS ON THE MODEL BRIDGE

Different kinds of vehicles are used in order to mimic a real world traffic pattern as seen traveling on the bridge. The vehicles chosen include: cars, vans, buses and large trucks. Varying vehicle speeds, weights and lengths are also used. The traffic is simulated on the bridge model and strain data is collected from the simulated gauges.

SHM data collected from the field for a vehicle traveling on a similar structure indicates that the generated strain pattern consist of a series of smooth peaks. These peaks are generated whenever the vehicle passes over a sensor. The data models used in this research artificially introduces these peaks to produce the individual simulated vehicle strain signatures. Thus the resulting strain peaks correspond to a vehicle axle passing over the simulated gauge.

The vehicle models that are used to generate simulated data are therefore largely based on vehicles axle characteristics. One of these characteristics includes forces exerted by each axle which determines the recorded strain value. The other characteristic is axle spacing which influences the distance between the strain peaks. A strain signature will therefore largely depend on the weight and length of the vehicle.

The strain data patterns can be divided into two main categories which include: car/van strain signatures and bus/large truck strain signatures. These patterns are divided into two categories based on similarities in their strain data patterns.

3.1.1 STRAIN SOFTWARE MODEL OF CARS AND VANS

Maximum strain is recorded by a gauge when a vehicle axle is right above it. This means that the number of peaks in strain data generated by a single gauge correlates to the number of axles of the vehicle. Also as described previously the distance between axles corresponds to the spacing between strain peaks. In the case of car and van vehicle group, which fall under FHWA class two and three respectively, have 2 axles per vehicle with a very short distance between them. Their simulated strain signature is therefore represented by a single peak.

The single peak vehicle strain pattern looks very similar to a Gaussian distribution. An un-normalized Gaussian distribution was therefore used to model strains generated by cars and vans. Equation 3-1 describes the vehicle model that was used:

$$StrainData(i) = w_k * e^{\frac{-(t_i - m)^2}{2\sigma^2}} \qquad (3\text{-}1)$$

In the above equation, w_k represents the weight of the vehicle, i represents the time sample in the data window, and m, which is the mean, establishes the relative placement of a pattern in an observation window. The variance or width of the pattern, σ, is based on the vehicles speed and its length. The different car characteristics used for the software model of a car are given in table 3-2. These characteristics were derived from actual vehicle characteristics shown in table 3-1. The actual vehicle characteristics that were used to determine the van software model characteristics are given in table 3-3. The software model characteristics for a van are given in table 3-4.

TABLE 3-1: CHARACTERISTICS OF DIFFERENT CARS

Types of Cars	Weights
Toyota Prius	1335kg
Pontiac Grand Prix	1461kg
Pontiac Coup	1598kg
Honda Accord	1630kg

TABLE 3-2: CHARACTERISTICS FOR THE SOFTWARE MODEL OF A CAR

Average weight	1450kg
2σ of weight	200kg
Average Length	4.5 meters
2σ of the length	0.5 meters

TABLE 3-3: CHARACTERISTICS OF DIFFERENT VANS

Types of Vans	Weights
Pontiac Montana	2075kg
GMC Savana	2276kg

TABLE 3-4: CHARACTERISTICS OF THE SOFTWARE MODEL OF A VAN

Average weight	2000kg
2σ of the weight	200kg
Average Length	4.5 meters
2σ of the length	0.5 meters

The above mentioned car/van model ensures that the respective vehicle strain signature is a smooth bell shaped pattern with the peak height corresponding to weight of the vehicle. The height is also the maximum strain that is generated by the gauge for that vehicle.

The weights of the cars and vans were varied and normally distributed with variance of 200kgs. The variation was introduced in order to incorporate the diverse characteristics of the different types of cars and vans. An average weight of 1450kgs and 2000kgs was used for cars and vans respectively. It can be determined from the chosen weight characteristics that the height of the peak in a van strain signature will be higher than the peak in the car strain signature because of their differing weights.

As mentioned previously the width of the vehicle strain signature, σ, is determined by the speed and length of the vehicle. A slower vehicle will result in a larger width because of its longer presence over the bridge and the sensor. On the other hand a faster vehicle will result in a narrower peak or a smaller width due to the shorter period of time that it is on the sensor. A vehicles length also plays a role on the patterns width, where a vehicle with a larger length will result in a larger width while a smaller length vehicle will generate a strain signature with a narrower width.

The average speed was chosen as 60km/hr with a variance of 25km/hr while the average length was chosen as 4.5 meters with a variance of 0.5 meters for both cars and vans. It can be determined from these chosen characteristics that the width of the signatures generated by cars and vans will be similar.

Figure 3-5 shows a graph of strain data generated by a single sensor when a car and a van passed over it. The graph shows that the van strain is indeed higher than the car strain.

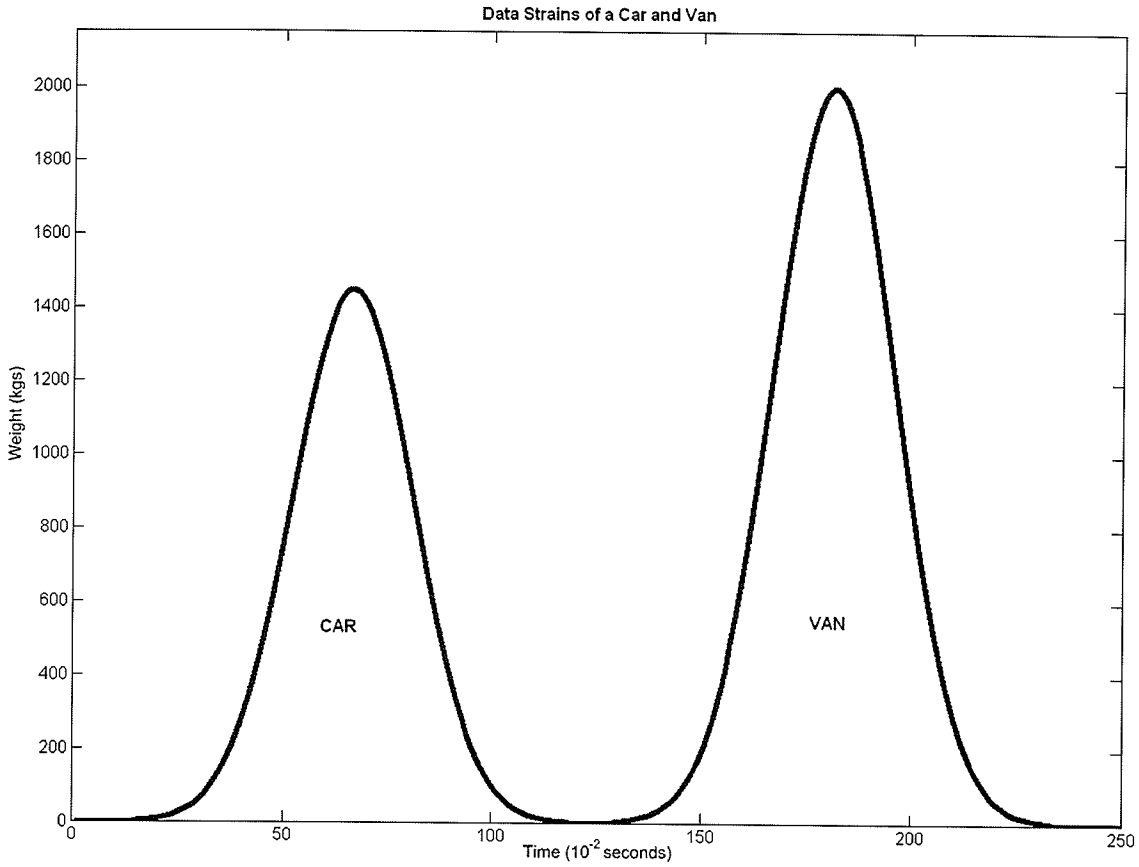

FIGURE 3-5: STRAIN SIGNATURES OF A CAR AND A VAN

3.1.2 STRAIN SOFTWARE MODEL OF BUSES

Buses, which would fall under FHWA class four, have a significantly larger distance between their axles when compared to cars and vans. They are also appreciably larger and heavier. The strain data generated by this vehicle is therefore modeled as having two prominent peaks representing the two axles.

Un-normalized Gaussians were again chosen to model the multiple peaks present in the strain pattern. Equation 3-2 represents the model that was used of a bus traveling over a gauge.

$$StrainData(i) = w_{k1} * e^{-\frac{(t_i - m_1)^2}{2\sigma_2^2}} + w_{k2} * e^{-\frac{(t_i - m_2)^2}{2\sigma_2^2}} + w_{k3} * e^{-\frac{(t_i)^2}{2\sigma_3^2}} \qquad (3\text{-}2)$$

The model equation consists of 3 Gaussians. The first two Gaussians represent the two main peaks in the pattern with their heights being w_{k1} and w_{k2}. Where:

$$w_{k1} = r * w_{k2} \tag{3-3}$$

Buses do not have an equal distribution of weight. The rear axle of the bus is usually heavier due to the engine. Therefore the bus was modeled as having one peak higher than the other. The r in equation 3-3 represents the ratio between the forces exerted by the front axle to that exerted by the rear axle. This ratio ensures that one of the peaks is higher than the other.

The means, m_1 *and* m_2, of the first two Gaussians determine the placement of the pattern in the observation window. They were chosen such that the distance of the peaks from the center of data pattern is the same. Therefore, if 0 is taken as the center of the observation window then:

$$m_1 = -m_2 \tag{3-4}$$

While the first two Gaussians were used for the two main peaks, the third Gaussian was used to partially fill the valley between these peaks. The mean is therefore 0 relative to the other Gaussian means so as to place it in between the two peaks.

Table 3-5 lists characteristics of the more common transit bus from New Flyer Industries Inc. For the bus model the average weight of a bus was assumed to be 18000kgs and the average bus length as 12 meters. The weight characteristic was varied so as to mimic real bus traffic. The average speed of the vehicle was also chosen as 60km/hr with a variance of 25km/hr. Table 3-6 lists the characteristics that were used for modeling buses in software.

TABLE 3-5: CHARACTERISTICS OF DIFFERENT BUSES

Bus Type	Weights
New Flyer	17000–20000kgs

TABLE 3-6: CHARACTERISTICS OF THE SOFTWARE MODEL OF A BUS

Average weight	18000 Kg
2σ of the weight	500 Kg
Average Length	12 meters
2σ of the length	0 meters

Figure 3-6 shows an example of a strain pattern generated when a bus traveled over a single sensor. The graph shows that the bus strain signature contains two prominent peaks with one being higher than the other.

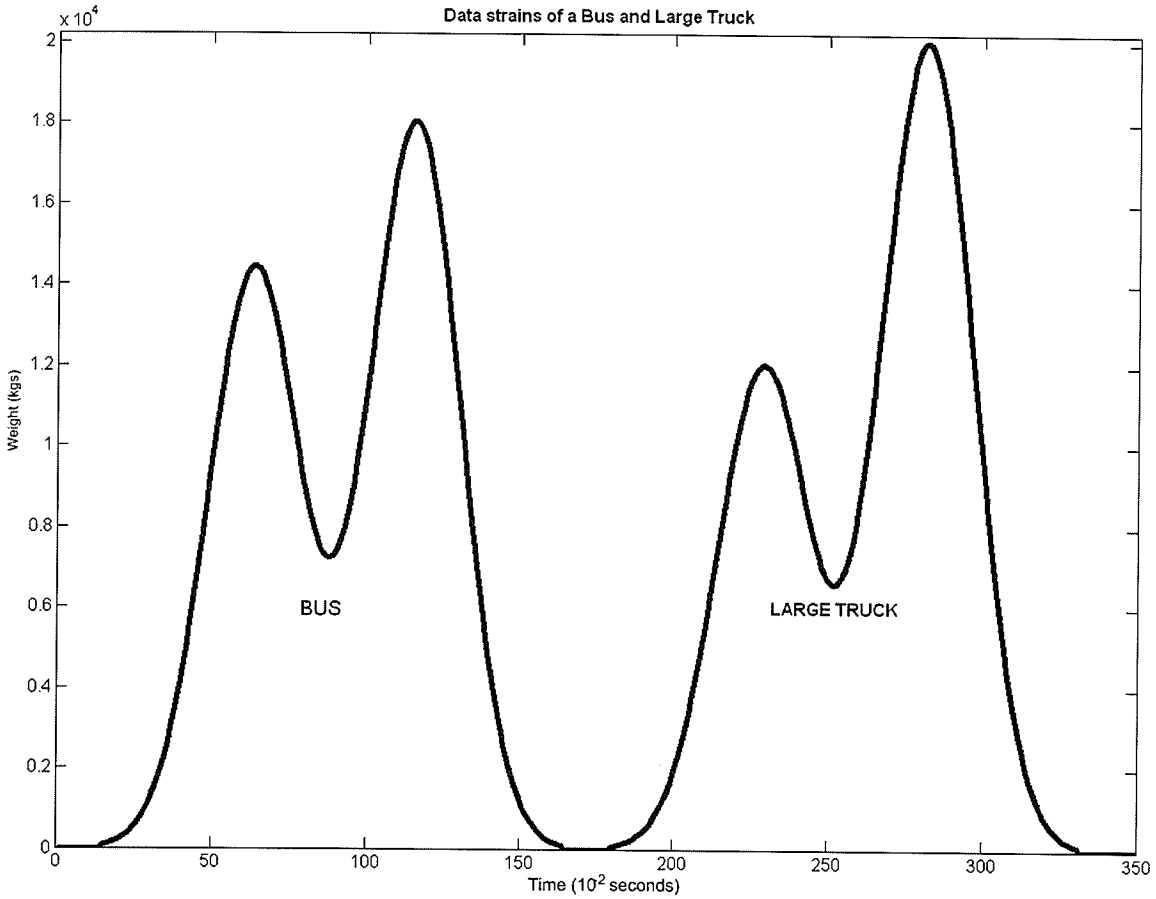

FIGURE 3-6: STRAIN SIGNATURES OF A BUS AND A LARGE TRUCK

3.1.3 STRAIN SOFTWARE MODEL OF A LARGE TRUCK

A large truck, which would fall under FHWA class eight, can be modeled very similarly to a bus, because it is heavy and also has a considerably larger distance between its axles when compared to cars and vans. The strain signature of a large truck is therefore modeled as having two prominent peaks. One feature that distinguishes a truck model from a bus model is the distribution of weight between the vehicle axles.

Modeling of a truck strain signature was again done using un-normalized Gaussians. Equation 3-5 represents the model that was used to generate strain data for a truck passing over a single sensor.

$$StrainData(i) = w_{t1} * e^{-\frac{(t_i - m_1)^2}{2\sigma_2^2}} + w_{t2} * e^{-\frac{(t_i - m_2)^2}{2\sigma_2^2}} \qquad (3\text{-}5)$$

The two Gaussians represent the two peaks present in the strain signature. The magnitude or the peak strains of the Gaussians is given in equation 3-6.

$$w_{t1} = r_t * w_{t2} \qquad (3\text{-}6)$$

The factor r_t in the equation above is the ratio of the force exerted by the front axles over the force exerted by the rear axles. A ratio of 0.6 was used to characterize the axle weight difference. The first Gaussian represents the heavier trailer portion of the large truck while the second Gaussian represents the lighter cab.

The means of the two Gaussians were chosen to be equidistant from the center of the pattern. Therefore if the vehicle is placed in the middle of window (at 0) then:

$$m_1 = -m_2 \qquad (3\text{-}7)$$

The following tables represent the characteristics of a large truck that were used in the above equation.

TABLE 3-7: CHARACTERISTICS OF DIFFERENT LARGE TRUCKS

Large truck Type	Weights
2 Axles	16400 Kgs
3 Axles	24300 Kgs

TABLE 3-8: CHARACTERISTICS OF THE SOFTWARE MODEL OF A LARGE TRUCK

Average weight	20000 Kgs
2σ of the weight	1000 Kgs
Average Length	12.5 meters
2σ of the length	0.5 meters

It was also assumed that the valley in a large truck pattern is more pronounced because of the different axle force ratio which is why the third Gaussian present in the bus model shown in equation 3-2, is absent in the large truck model.

An average large truck weight of 20000kgs with a variance of 1000kgs was used along with an average truck length of 12.5 meters with a variance of 0.5 meters. An average speed of 60km/hr and a variance of 25km/hr was also used for the large truck. Figure 3-6, on page 26, shows the strain pattern generated using the above model when mimicking a large truck traveling over a single sensor.

3.2 GENERATION OF DATASETS FOR LEARNING, TESTING AND VALIDATION

The above mentioned car, van, bus and large truck models were used to generate datasets in order to train, test and validate the neural system classifiers. Different vehicles strains generated using varying vehicle traits and speeds were randomly placed in the datasets to mimic a real world characteristic. The dataset was also generated in such a way that it contained a mixture of all the vehicle types. The speeds of the vehicles were varied and randomly distributed so as to give a more real-world type simulation. The weight and length of each vehicle was also randomly picked using the vehicle's respective characteristics as mentioned previously.

The four graphs in figure 3-7, figure 3-8, figure 3-9, and figure 3-10 illustrate the data captured at the same time over a period of 4 seconds from all four sensors. The strain pattern shifts from sensor to sensor. The first sensor sees the whole strain pattern while the rest of the sensors see

partial strain patterns. This is because the sensors are placed apart on the bridge and therefore each sensor sees the vehicle at different points in time.

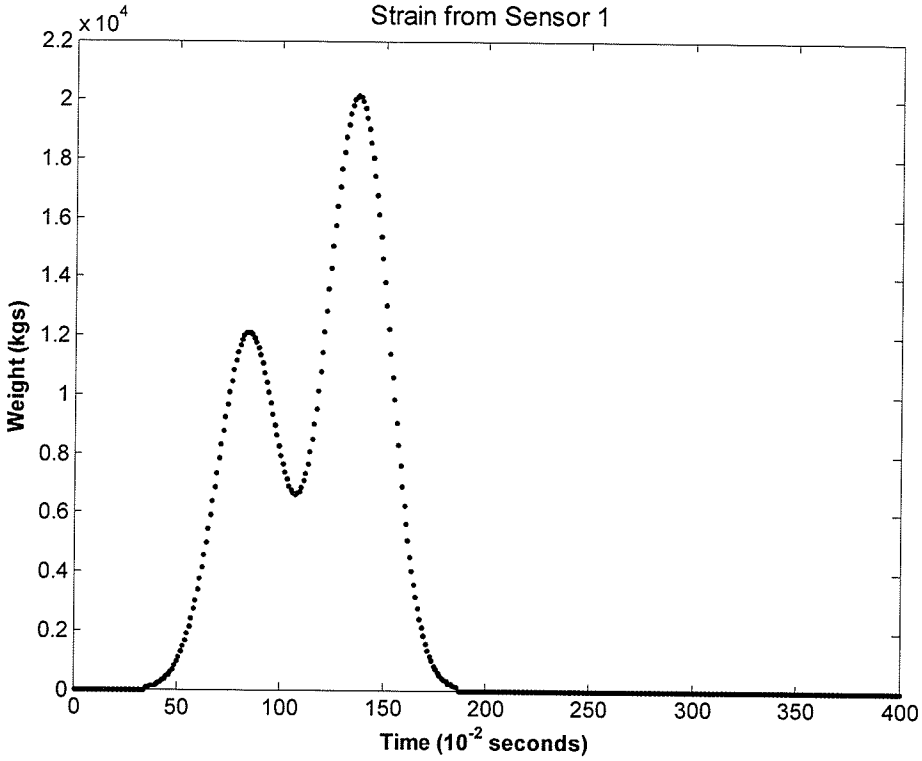

FIGURE 3-7: STRAIN SIGNATURE FROM SENSOR 1

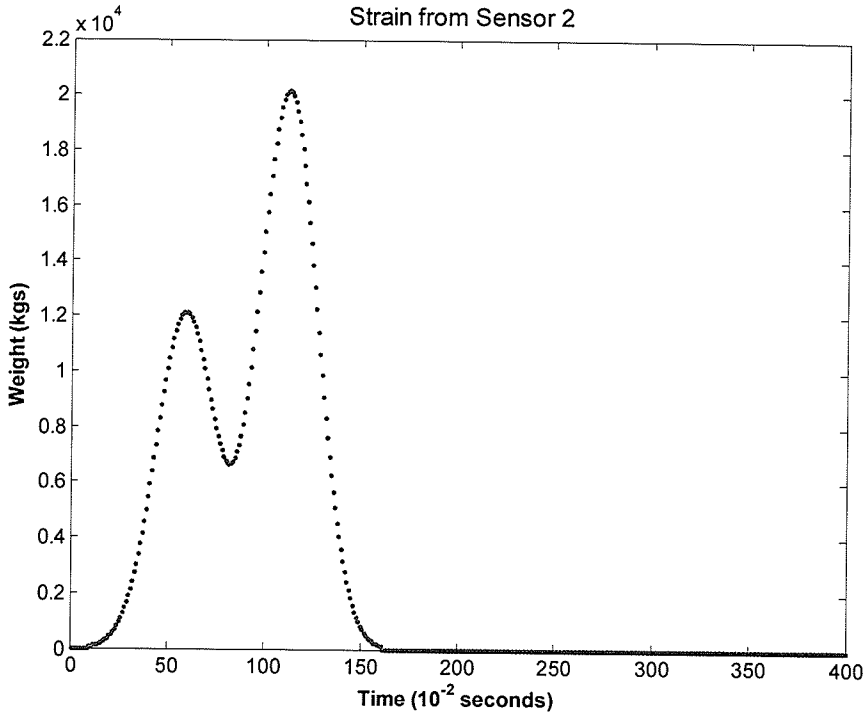

FIGURE 3-8: STRAIN SIGNATURE FROM SENSOR 2

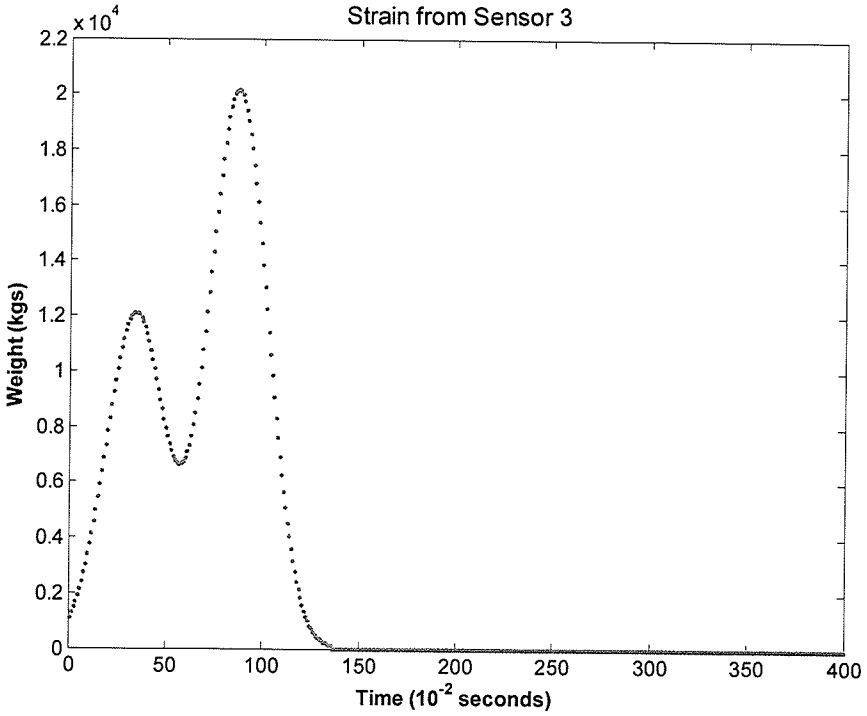

FIGURE 3-9: STRAIN SIGNATURE FROM SENSOR 3

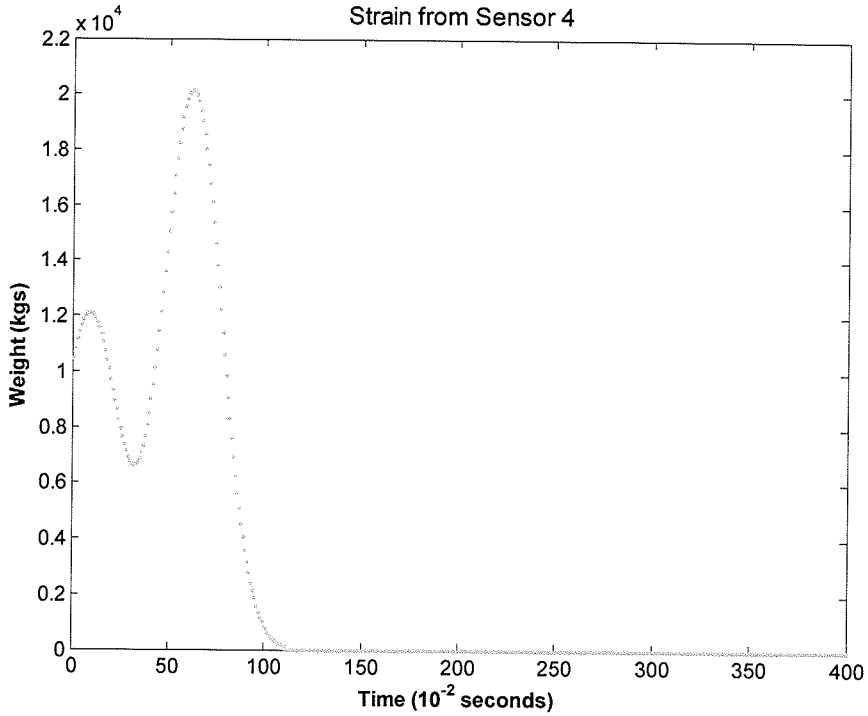

FIGURE 3-10: STRAIN SIGNATURE FROM SENSOR 4

There were two kinds of datasets that were primarily produced. The first one was a sparse dataset which had isolated vehicle strains while the second one was generated in such a manner that it had complete data strains for two vehicles with overlapping time windows.

3.2.1 SPARSE DATASET

The sparse dataset was used for both training and testing purposes. It was generated in such a manner that a four second window contained one and only one vehicle strain pattern. The pattern was randomly placed in this window to give it a real-world characteristic. The training of the neural system was mainly done using this type of dataset because sparse events are more straightforward to learn. Figure 4.5 shows this kind of dataset from a single bridge sensor.

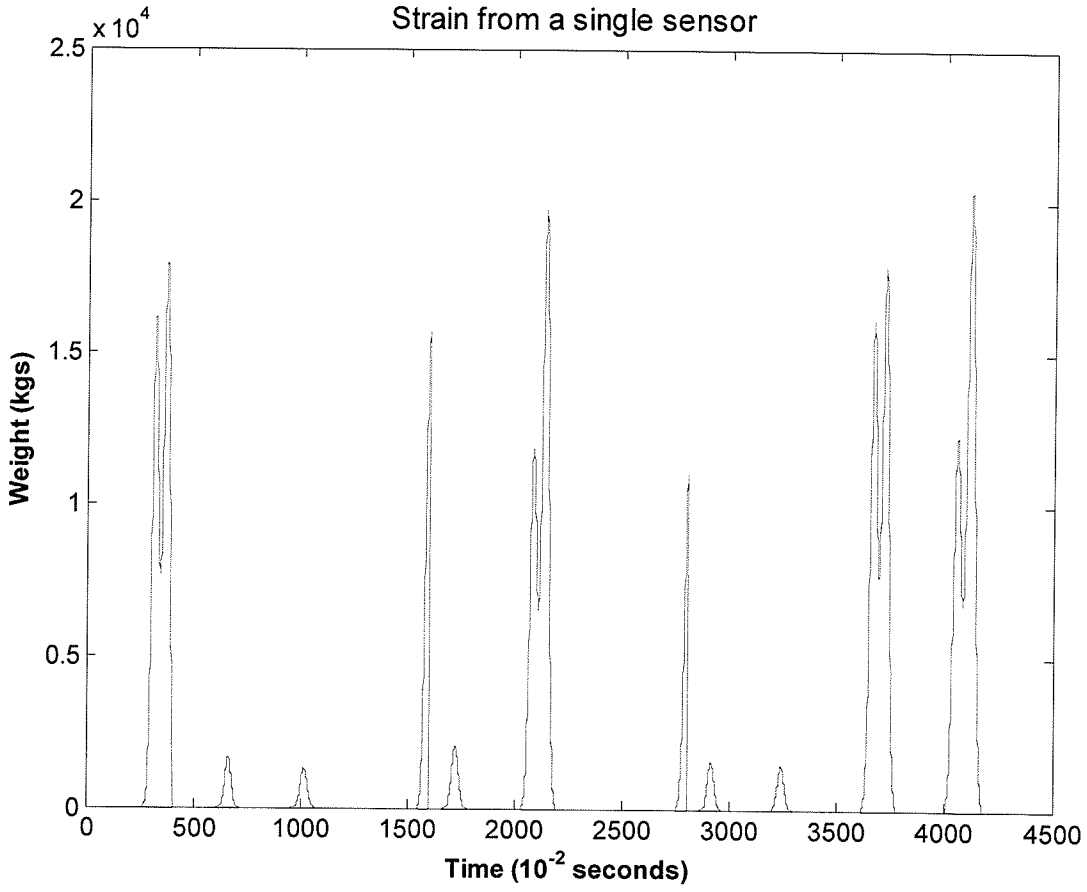

FIGURE 3-11: SPARSE DATASET FROM A SINGLE SENSOR

3.2.2 OVERLAPPING DATASET

The second type of data strain pattern set that was generated was the overlapping strain pattern. This type of dataset was used to model a real world setting where vehicles are sometimes very close together when traveling. The patterns in this dataset sometimes overlapped each other blurring the distinction between the strain patterns. Figure 3-12 illustrates the sensor data from all the four sensors placed on a single time scale. It can be seen in this figure that the same strain pattern is seen by all the sensors but occurring at differing times.

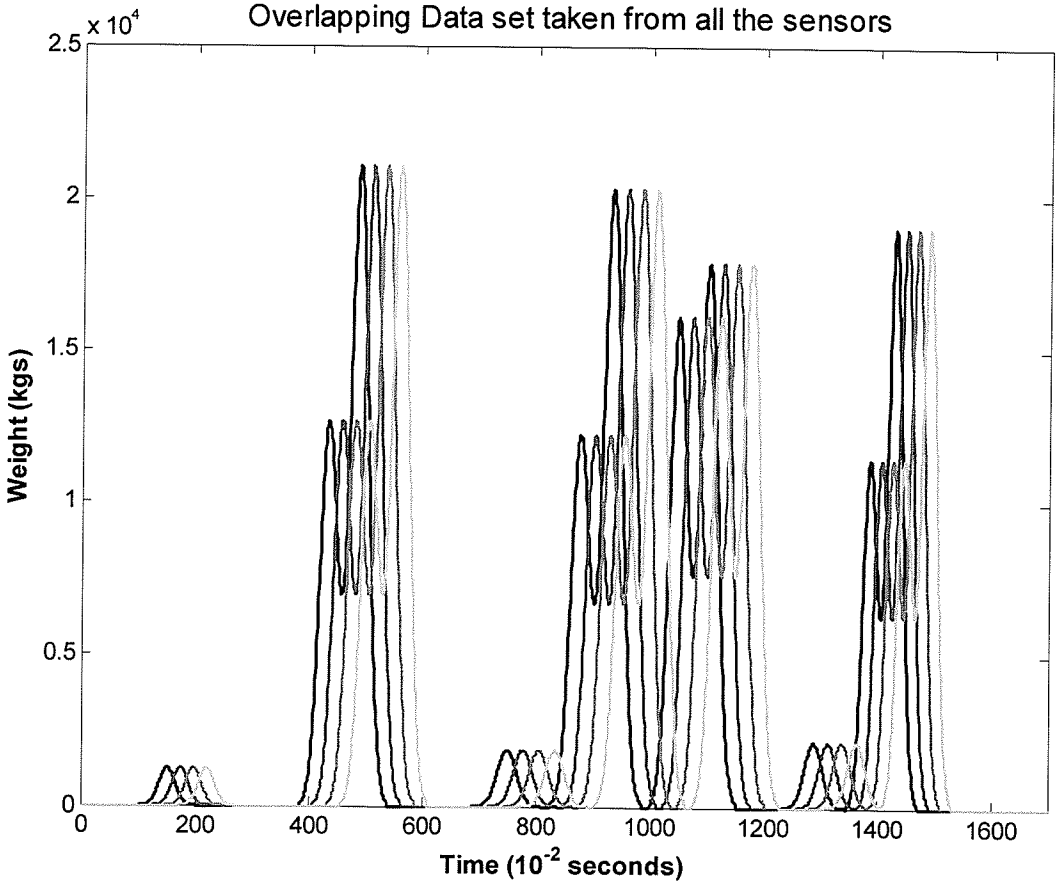

FIGURE 3-12: OVERLAPPING DATASET FROM ALL 4 SENSORS

4 NEURAL CLASSIFIERS

The approach undertaken in this work utilizes artificial neural networks (ANN) to classify individual traffic events recorded within SHM data. The ANN classifiers incorporate a combination of supervised and unsupervised neural learning methods to perform event identification and classification.

The main objective of these classifiers is to identify vehicle strains from a continuous stream of data without having to segment out individual vehicle responses for the purpose of classification.

The unsupervised learning method used to train some of the networks makes use of frequency sensitive competitive learning (FSCL) algorithm which was described in chapter 2. Also as described in chapter 2, supervised learning is achieved by providing the learning algorithm with input data and the desired output.

4.1 ARTIFICIAL NEURAL NETWORK VEHICLE CLASSIFIER SYSTEM

The first neural classifier system that is employed in this study consists of two layers of artificial neurons. The main purpose of using two layers is such that one of the ANN layers can behave as a feature extractor while the second behaves as an event identifier. Both of these stages are to be trained using data generated from the previously described vehicle data models. Once trained, a single four second window of data, taken from all four sensors, forms input to the system. The neural classifier is illustrated in figure 4-1 below.

The first neural network layer, which is trained using the FSCL algorithm, is the one that acts as a feature extractor. The main purpose of this layer is to discover regularities within the input sensor measurements. The second layer, trained using supervised learning, acts as the event classifier whose main purpose is to group SHM events into classes. The data is first presented to the feature extractor. The outputs of this layer go through a scaling process before being presented to the next neural layer. The data is then processed by the second layer whose outputs are in terms of percentages of each of the four classes. The percentages represent the similarity between the input and any particular class. The class with the highest percentage is usually the source of the SHM event.

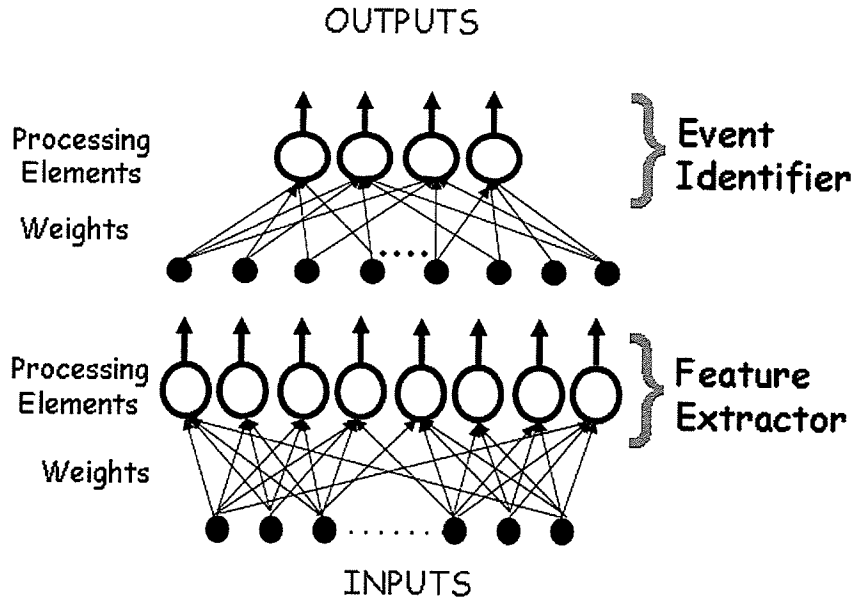

FIGURE 4-1: ARTIFICIAL NEURAL NETWORK VEHICLE CLASSIFIER SYSTEM

Training of the neural system takes place by training the two layers of neurons independently. The feature extractor is trained before the event classifier. Training is done in stages in order to ensure that the event identifier gets input from a trained feature extractor in order to maintain consistency in input data. The dataset containing isolated events is used during this training. This training dataset is a combination of the strain measurements as well as their causes for example car, van, bus or large truck.

Training of the feature extractor is accomplished by feeding it with windows of raw sensor data from all four sensors and training the network on this data using the FSCL algorithm. The FSCL algorithm performs competitions among neurons and modifies the neuron weight vectors depending on the cost and frequency of wins. The purpose of training the first stage in this manner is to ensure that the data patterns with similar features can be grouped together. Once the training process is completed, the feature extractor produces output activations from each of its neurons in response to a given input pattern. The magnitude of each neuron's response to an input is directly related to the similarity between that input and the individual neuron's learned weights. The neuron with the largest output activation is necessarily the neuron which best matches features of the current input dataset.

Determining the number of neural processing elements (neurons) to use in a given layer of a neural classifier is not a straightforward procedure. In order to select a suitable size for the first feature extraction layer, networks of 7, 8 and 9 neurons were trained and subsequently evaluated based on the average error magnitude produced by the layer when representing the input data. It was found that a network with 8 or 9 neurons gave the lowest average error during training and therefore the use of either would be suitable. Similarity in the error for networks of 8 and 9 neurons means that the extra neuron is redundant in this application and thus can be eliminated. Therefore, a total of 8 neurons are used in this initial stage of processing.

The output activations of the first layer are then passed through a radially symmetric, non-linear function (an exponential) in order to scale the outputs before they are presented as inputs to the next neural layer.

The second layer of the vehicle classifier system, otherwise known as the event identifier, is trained next. The main purpose of this layer is to use the data generated by the feature extractor and make decisions about the vehicle group that caused the data pattern. The layer therefore consists of four neurons, each of which corresponds to the four vehicle types. The neurons are trained using a supervised learning approach. Training of the neurons is done by presenting the learning algorithm with scaled outputs (from the non-linear function) of the first layer together with the desired output. The desired output in this case is the vehicle type classification of the event. The neurons are trained using the presented data and form a relationship between the

inputs and the desired output classifications. The second layer is thus trained to act as an event identifier, classifying input events using the data presented to it from the preceding feature extraction layer. Training of the neural classifier was done using 3000 isolated strain measurements.

To avoid misclassifications resulting from partial event profiles being present within an observation window, the neural classifier makes use of a majority rule at the output of the second layer. The majority rule gives a more precise result by combining the evidence produced by the classifier over time as the same vehicle passes through the observation window. Five consecutive output decisions are considered in making the final vehicle type classification.

Table 4-1 illustrates the result of incorporating the majority rule. The table contains output neuron activations from the event identifier stage. For the case of window 1, the neural classifier identified a van as the most likely cause of the strain shown by the highest activation. The neural classifier then goes on to identify a van, van, car and car as the likely cause of the strains present in windows 2, 3, 4, and 5. The majority rule algorithm takes the results from the 5 windows and results the van as the winner of the competition.

TABLE 4-1: EXAMPLE OF THE MAJORITY RULE BEING APPLIED TO VEHICLE TYPE DECISIONS.

Window	Car	Van	Bus	Large Truck	Decisions
1	0.88	0.97	0.55	0.61	Van
2	0.89	0.97	0.55	0.60	Van
3	0.92	0.94	0.52	0.58	Van
4	0.97	0.94	0.52	0.57	Car
5	0.94	0.89	0.56	0.61	Car

4.1.1 RESULTS

Once the vehicle classifier system was trained, it was assessed based on its ability to correctly or partially identify the vehicles which caused the event. The network was first tested using a sparse dataset. The results are tabulated in table 4-2 and table 4-3.

TABLE 4-2: ORIGINAL VEHICLE EVENTS FOR THE ANN CLASSIFIER

Original event causes	
Vehicle Group	**Total**
Car	698
Van	807
Bus	728
Large Truck	768

TABLE 4-3: RESULTS OF CLASSIFICATION BY ANN CLASSIFIER

Identified by Neural Classifier	
Vehicle Group	**Total**
Car	768
Van	737
Bus	598
Large Truck	898

The network was tested using 3001 data patterns of the sparse dataset with the absence of the majority rule. It can be seen from the table that imperfect classifications were made. Further data analysis revealed the misclassifications and these results are tabulated in table 4-4 below.

TABLE 4-4: ANN CLASSIFIER SYSTEM FUTHER DATA ANALYSIS

		Identified by the neural classifier system			
		Car	Van	Bus	Large Truck
Event Causes	Car	552	146	0	0
	Van	216	591	0	0
	Bus	0	0	288	440
	Truck	0	0	310	458

Of the original 698 cars, 552 were correctly identified while the rest were identified as vans. In the case of vans, 591 were correctly identified while the 216 were wrongly identified as cars. The classification was worst for buses, where out of the original 728 buses, only 288 were correctly identified. As for the case of large trucks, 458 of them were correctly identified. Therefore the system was able to achieve a success rate of 63%.

Classification errors which account for the rest of the 17% are a result of partial vehicle strain patterns captured in the observation window. For example the strain pattern shown in the figure 4-2 belongs to a van but it was wrongly identified as a car. As can be seen in the figure, strain sensor 1 sees the peak of the data pattern while sensors 2 and 3 see only a partial strain pattern. The observation window doesn't capture any data from the 4^{th} sensor. This is due to the fact that the vehicle had not reached the sensor during that time period. The artificial neural classifier therefore identified the partial data pattern as belonging to a car.

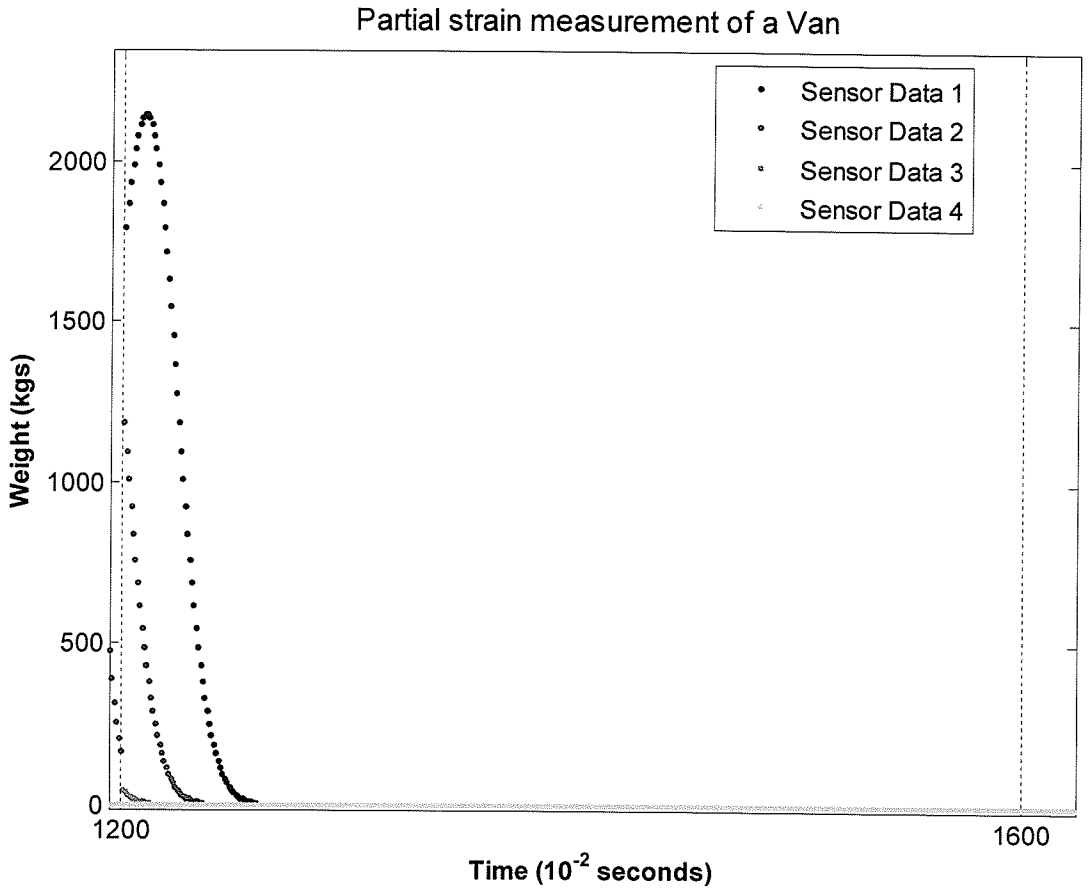

FIGURE 4-2: PARTIAL STRAIN PATTERN OF A VAN

4.2 ARTIFICIAL NEURAL NETWORK VEHICLE CLASSIFIER SYSTEM WITH SCALING

One of the major shortcomings with the initial classifier was its inability to adequately differentiate between cars and vans, and between buses and large trucks, even though they have quite different features. Further analysis of the classification process revealed that the misidentification was happening at the feature extraction layer. It was determined that the training process of the first artificial neural network layer was assigning more neurons to inputs with larger magnitudes than to the lower ones. This was mainly because of significant differences in magnitude between the car/van and bus/large truck strains. To prevent the training process from giving preference to higher magnitude strains, an extra stage of data pre-processing was added to the classification system.

The added pre-processing stage to the original vehicle classifier system executes a scaling operation on any new input data before it is presented to the neural network. All data is scaled to the maximum allowable input value within that particular measurement window. The added processing step puts all the strain patterns at the same peak magnitude with the intention of then making it easier to distinguish the overall shape of the underlying strain signature. Information concerning the magnitude adjustment applied when performing the scaling is not discarded. It is passed on to the event identification layer as a further supplementary input. The second layer is therefore presented with output activations from the feature extractor as well as information concerning the maximum magnitude from the data pre-processing stage. Figure 4-3 shows the resultant neural network system.

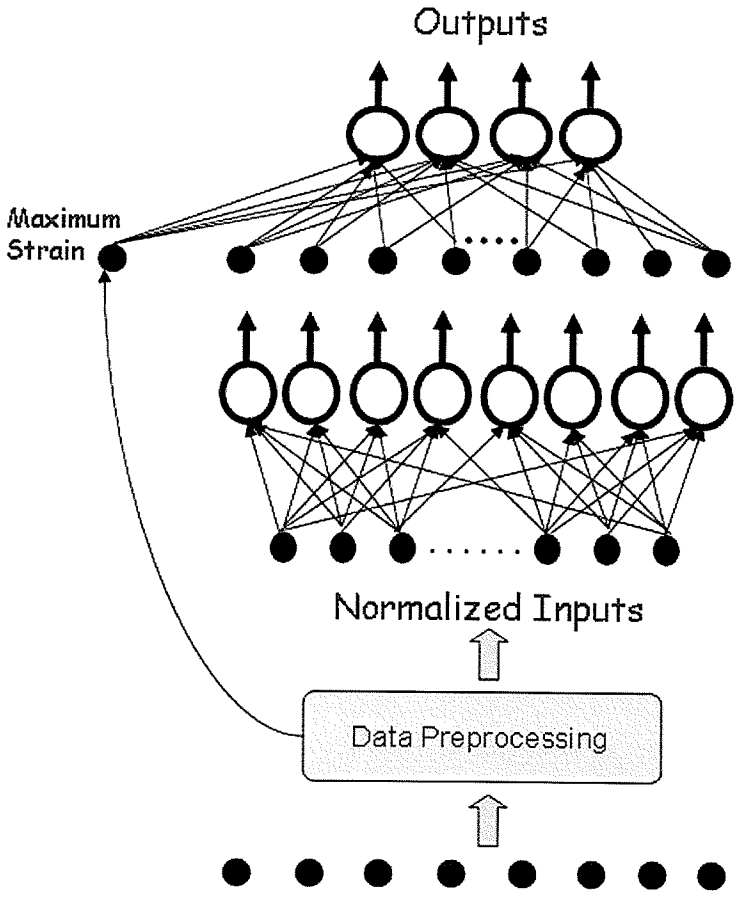

FIGURE 4-3: ARTIFICIAL NEURAL NETWORK VEHICLE CLASSIFIER SYSTEM WITH INPUT SCALING

Training of the following neural classifier system is also performed in two phases. The first phase of training, trains the feature extraction layer while the second phase consists of training the classification layer.

Training of the first layer of the neural network is accomplished by the FSCL algorithm. Training begins by presenting raw strain measurements from the isolated dataset, in windows of 4 seconds, to the neural system. The input scaling stage scales these inputs as part of the data pre-processing and presents them to the FSCL algorithm in order to train the feature extractor. The maximum magnitude extracted during scaling is stored to be later used to train the second layer of the ANN. The FSCL algorithm modifies the weight vectors of the first layer while performing competitions amongst neurons. The training is deemed completed when the average mean squared error approaches a constant. Once the training is concluded, the neuron that gives the best activation is the neuron whose weight vectors best match features of scaled strain signatures in the input data window.

The second stage of the ANN can now be trained using the output activations from the previous layer of the ANN. The output activations from the first layer ANN are passed through a non-linear function, in this case an exponential, to further scale it. The weight vector was randomly initialised to the first vector out of the exponential. Training of the second layer was performed using the supervised learning algorithm by providing it with scaled output activations, maximum allowable magnitude from the scaling stage, and the likely cause of the input strain. Again the training was deemed completed with the average error reached a constant.

4.2.1 RESULTS

Just like the original vehicle classifier system, the following classifier system was tested using both the isolated dataset and the overlapping dataset. The following results were obtained when the classifier was tested using the isolated dataset:

TABLE 4-5: ORIGINAL VEHICLE EVENTS FOR THE ANN CLASSIFIER WITH SCALING

Original event causes	
Vehicle Group	**Total**
Car	728
Van	770
Bus	736
Large Truck	767

TABLE 4-6: RESULTS OF CLASSIFICATION BY ANN CLASSIFIER WITH SCALING

Identified by Neural Classifier	
Vehicle Group	**Total**
Car	803
Van	695
Bus	847
Large Truck	656

The network was tested using 3001 data patterns of the sparse dataset with the absence of the majority rule. Further data analysis indicated that there were misclassifications. Table 4-7 below tabulates those results.

TABLE 4-7: ANALYSIS OF RESULTS FROM ANN CLASSIFIER SYSTEM WITH SCALING

		Identified by the neural classifier system			
		Car	Van	Bus	Large Truck
Event Causes	Car	727	1	0	0
	Van	76	694	0	0
	Bus	0	0	710	26
	Truck	0	0	137	630

It is seen from the table that the classifier system was able to correctly identify 727 cars of the original 728 car strains. The classifier system also identified 694 vans, 710 buses and 630 large trucks correctly. The table therefore shows that the normalised neural classifier has a detection rate of 92.01% which is significantly greater than the original classifier system. The remaining 8% of misclassifications were found to be happening when the vehicle was either just leaving the observation window or just entering it.

The next test that was performed was using the overlapping dataset. In this situation two vehicles are present in the same observation window imitating two vehicles travelling in close proximity of each other. A snapshot of some results from analysis of 101 vehicles is captured in table 4-8.

TABLE 4-8: VEHICLE CLASSIFICATION BY AN ANN WITH SCALING WHEN USING OVERLAPPING DATASET

		Output data from neural classifier system with scaling				
Vehicles present in window		**Window 1**	**Window 2**	**Window 3**	**Window 4**	**Window 5**
3	2	3	3	3	2	2
2	1	2	2	2	2	1
0	2	0	2	2	2	2
2	2	2	2	2	2	2
1	3	1	2	3	3	3
1	2	1	2	2	2	2
0	2	2	2	2	2	2
1	1	1	1	1	1	1
0	3	0	0	2	3	3
3	3	3	3	3	3	3

In table 4-8, a car, van, bus and a large truck are represented by 0, 1, 2, and 3 respectively. The first two columns consist of the vehicles that are present in the observation window. For example in the first case, a large truck and a bus were present in the data window. The neural network analysis of 5 moving windows is given in the rest of the columns. The neural classifier system in this case identified the first three of the overlapping windows as containing strains caused by large trucks while the last two as those caused by a bus. An overlapping factor of 0.25 seconds was used between the windows.

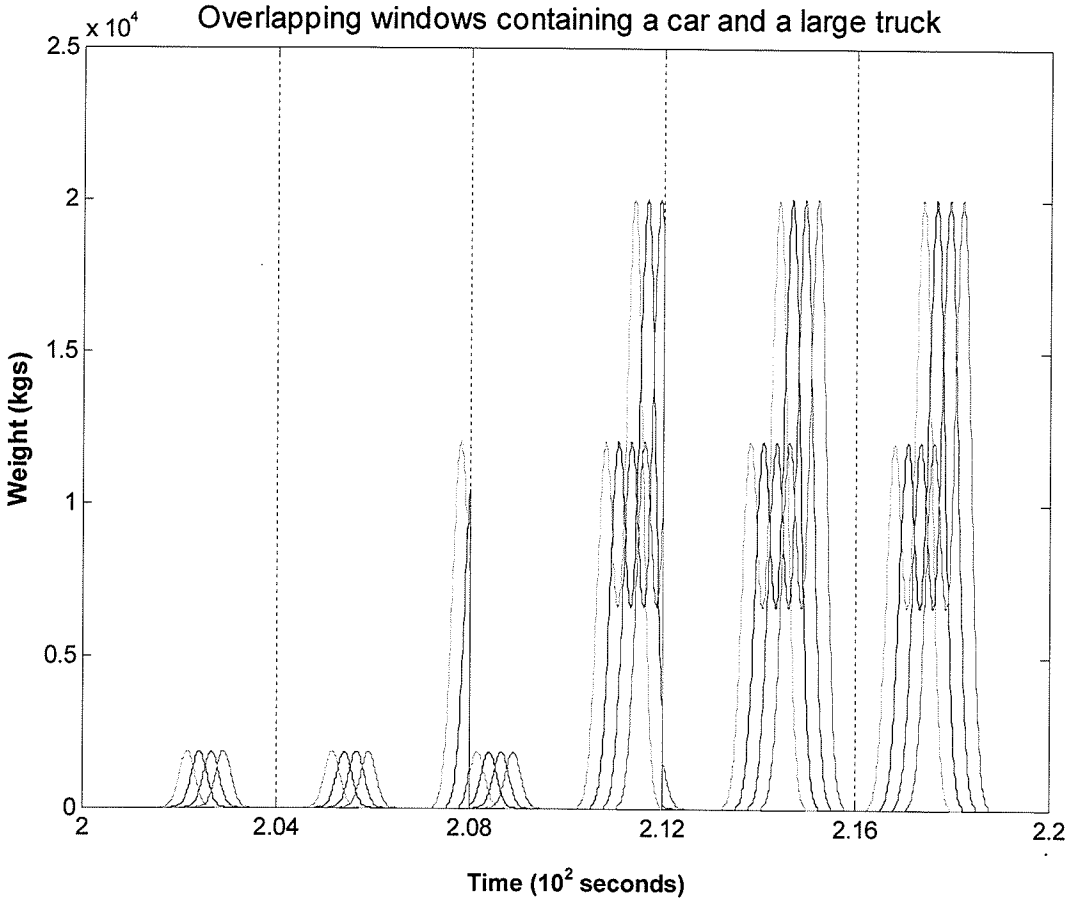

FIGURE 4-4: STRAIN SIGNATURE OF OVERLAPPING CAR AND LARGE TRUCK STRAINS

The highlighted row in table 4-8 indicates that a van and a large truck are present in the overlapping windows. Figure 4-4 shows the five overlapping observation windows that were analysed by the neural classifier. From the table it can be seen that the network identifies the first window as being a van. This makes sense given that only the van signature is visible in the window. The network identifies the 2nd window as that belonging to a bus even when the van is present and seen by all the four sensors. The network seems to pick the higher magnitude strain during the identification process. The network classifies the rest of windows as those containing a large truck. This reveals that the classifier does not do a good job of identification when strains of different magnitudes are present in the same observation window.

Analysis revealed that the scaling stage reduced the already small strain magnitude of a car/van in an observation window containing a larger magnitude vehicle to one with a very small strain.

The classifier system was therefore not able to detect them properly. The result of this degradation also caused some strains that are present in the window for longer periods of time than the higher magnitude strains to not be detected.

4.3 DISTRIBUTED ARTIFICIAL NEURAL NETWORK VEHICLE CLASSIFIER SYSTEM

One of the shortcomings observed with the previous system is the preference to select an input pattern classification which corresponds only to the higher magnitude response present in the observation window. The presence of a second vehicle with lower magnitude response is effectively lost. For example, when a car and truck are both present, the resulting classification will favour the truck, rather than identifying the presence of both vehicles together.

In an effort to correct for this problem, inputs from each of the four sensors were separated and provided to their own individual sub-network. This is illustrated in figure 4-5. A single sub-network consists of a feature extractor and an event identifier. Data from each sensor is first normalized and then presented to the feature extractor layer of each subsystem. Scaled outputs from the feature extractor are then presented to the event identifier along with the maximum strain extracted during the pre-processing stage. Feature extractor output scaling is done using an exponential. The classifications from each of the four event identifiers are then forwarded to the final layer for processing. A final classification decision is then made using the maximum number of wins from the four subsystem classifications. Training of the subsystems is accomplished by first training only a single subsystem and then replicating the same trained sub-network for each the four sensors. The final output stage was also trained using a supervised learning algorithm.

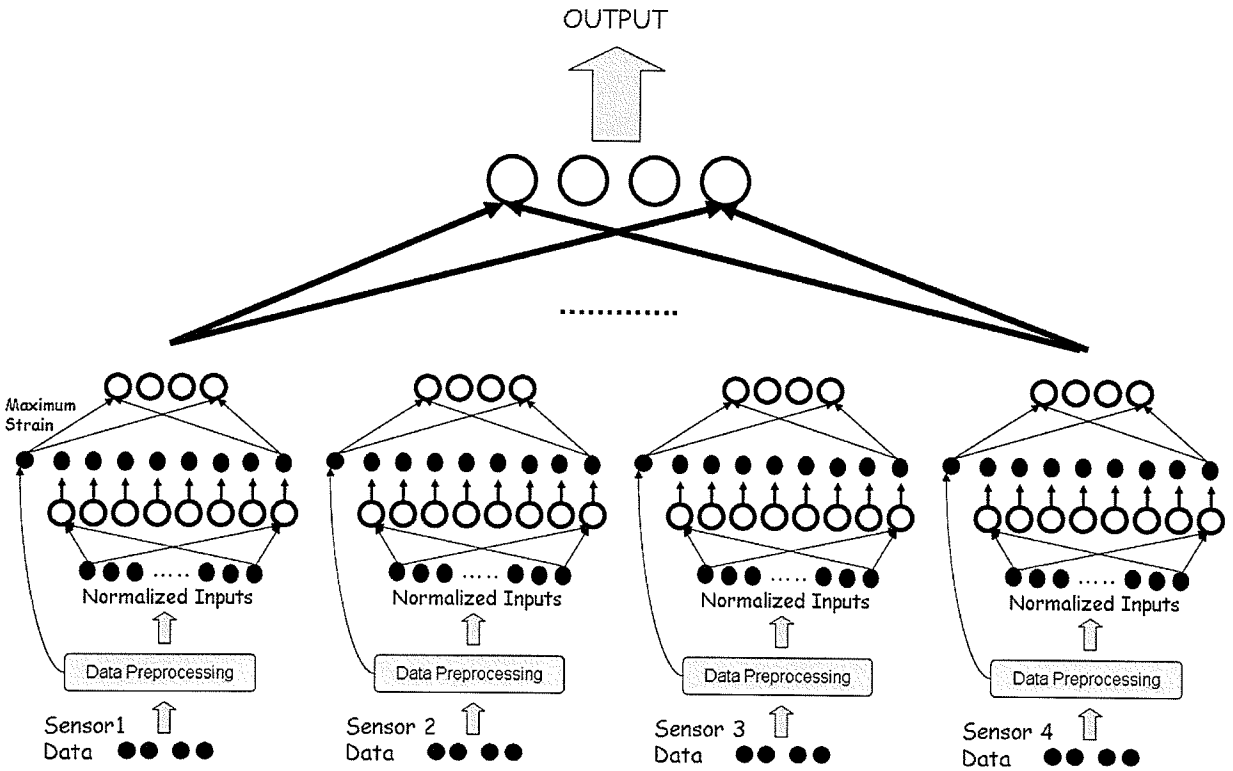

FIGURE 4-5: DISTRIBUTED ARTIFICIAL NEURAL NETWORK VEHICLE CLASSIFIER SYSTEM

The single NN sub-system was trained in a similar fashion to the neural classifier system with scaling. The feature extractor was trained using the FSCL algorithm followed by the event identifier which was trained using supervised learning algorithm. The training was done using the isolated vehicle dataset. Once the training was done, the trained sub-system was replicated to form the four individual sub-systems. Training of the final layer was then performed.

4.3.1 RESULTS

The following NN classifier system was first tested using the isolated dataset. It was presented with the same dataset that was used to test the neural classifier system with scaling. As mentioned previously, this dataset contains 3001 isolated vehicle signatures. Some of these signatures are partial, which happens when a vehicle is just leaving or entering the observation window. The classification results are shown in table 4-9 and table 4-10 below.

TABLE 4-9: ORIGINAL VEHICLE EVENTS FOR THE DISTRIBUTED ANN CLASSIFIER

Original event causes	
Vehicle Group	**Total**
Car	728
Van	770
Bus	736
Large Truck	767

TABLE 4-10: RESULTS OF CLASSIFICATION BY THE DISTRIBUTED ANN CLASSIFIER

Identified by Neural Classifier	
Vehicle Group	**Total**
Car	871
Van	659
Bus	840
Large Truck	631

It can be seen from the tables that the system was not able to make perfect classifications. The classifier seems to classify more strain signatures as being cars or buses. Further analysis of the output data was done and is presented in table 4-11.

TABLE 4-11: ANALYSIS OF RESULTS FROM DISTRIBUTED ANN CLASSIFIER

		Identified by the neural classifier system			
		Car	Van	Bus	Large Truck
Event Causes	Car	708	20	0	0
	Van	163	607	0	0
	Bus	0	0	671	65
	Truck	0	32	169	566

It is seen from the table that the classifier system was able to correctly identify 708 cars of the original 728 car strains. The classifier system also identified 607 vans, 671 buses and 566 large trucks correctly. The table therefore shows that the distributed neural classifier has a detection rate of 85.04% which is significantly greater than the original classifier system but lower than the neural classifier system with scaling.

The remaining 15% of misclassifications were found to be happening when there were partial vehicle signatures present in an observation window. The distributed neural classifier system analyses individual sensor strains separately, therefore if a strain pattern is partially present for a single sensor, it might not be present for another sensor in that observation window. The sub-systems will therefore give different classifications thus giving a wrong final decision.

A second set of tests that were done used the overlapping dataset. The same dataset that was used in the neural classifier with scaling was used here. The dataset contains two vehicle strain patterns in one observation window. This simulates vehicles travelling in close proximity of each other. Snapshot of some results from analysis of 101 vehicles is captured in table 4-12.

TABLE 4-12: VEHICLE CLASSIFICATION BY DISTRIBUTED ANN CLASSIFIER USING OVERLAPPING DATASET

		Output data from neural classifier system with scaling				
Vehicles present in window		**Window 1**	**Window 2**	**Window 3**	**Window 4**	**Window 5**
3	2	3	3	3	2	2
2	1	2	2	2	1	1
0	2	0	2	2	2	2
2	2	2	2	2	2	2
1	3	1	1	3	3	3
1	2	1	2	2	2	2
0	2	1	2	2	2	2
1	1	1	1	1	1	1
0	3	0	0	2	3	3
3	3	3	3	3	3	3

In table 4-12, a car, van, bus and a large truck are represented by 0, 1, 2, and 3 respectively. The first two columns consist of the vehicles that are present in the observation window. The neural

network analysis of 5 moving windows is given in the rest of the columns. Again, an overlapping factor of 0.25 seconds was used between the windows.

The highlighted row in table 4-12 indicates that a van and a large truck are present in the overlapping windows. Figure 4-4 shows the five overlapping observation windows that are analysed by the neural classifier. From the table it can be seen that the network identifies the first window as being a van. This makes sense given that only the van signature is visible in the window. The network identifies the second window as a van even when a partial large truck signature is visible by at least three of the four sensors. This shows that the network picks the signature that is visible by most of the sensors as opposed just picking the higher magnitude strain during the identification process. The network classifies the rest of windows as those containing a large truck. This reveals that the classifier does a better job of identification when strains of different magnitudes are present in the same observation window.

The distributed ANN classifier system had an identification success rate of 85% for an isolated data set and it was able to better detect at least one of the vehicles when two vehicle strains were present in the same observation window when compared to the ANN classifier with scaling.

5 CONCLUSION AND FUTURE RECOMMENDATIONS

This thesis examined the use of neural networks to identify and classify interesting or novel events from SHM data. The main objective of the research was to identify vehicle strains from a continuous stream of data without having to segment out individual vehicle responses for the purpose of classification. It also looks at the robustness of the classification system to varying vehicle speeds, weights, and lengths.

A simple software model of the Red River North Perimeter Bridge was created to simulate vehicles such as cars, vans, buses and large trucks travelling over the bridge. Strain data collected from the field indicated that the generated SHM vehicle strain pattern consists of a series of smooth peaks. The peaks are generated whenever a vehicle axle passes over a sensor. The software vehicle models therefore used this characteristic to mimic a vehicle travelling over a bridge. Two types datasets were generated. One was an isolated dataset which consisted of data windows having only one randomly placed vehicle strain signature. The other dataset was the overlapping dataset. These strain signatures were later used to train and simulate the neural systems.

Three neural systems, consisting of multiple artificial neuron layers trained using unsupervised and supervised learning algorithms were investigated. The first artificial neural network vehicle classifier system consisted of two layers of artificial neurons. The first layer, which was trained using FSCL, acted as a feature extractor and the second layer, which was trained using supervised learning, acted as an event identifier. This system had a success rate of 63% for an

isolated dataset containing 3001 data strain patterns. The misclassifications were a result of partial strain patterns captured.

The second neural system that was investigated was the artificial neural network vehicle classifier system with scaling. This network had the same structure as the previous artificial neural network vehicle classifier system except for an additional data pre-processing stage. The added scaling stage took the input data and scaled it to the maximum allowable input value in the measurement window. The scaled data was then passed onto the feature extractor and finally to the event identifier. The system produced an identification success rate of 92.01% for a dataset containing 3001 isolated vehicle strain patterns. The 8% identification error was found to be because of partial strains present in the observation windows. A second test using an overlapping dataset was performed. This dataset consisted of two vehicles in the same observation window. It was found that the classifier system was able to identify at least one of the vehicles in most cases. The results also indicated that the classifier system gave preference to a higher partial strain signature even when a full lower magnitude strain signature is present in the observation window.

The third system that was investigated was the distributed ANN vehicle classifier system. This system performed a sensor independent strain data analysis. The classifier consisted of four subsystems each of which had a scaling stage, a feature extractor and an event identifier. Each subsystem took the sensor data, scaled it and passed it on to the feature extractor and finally the event identifier. A final stage took outputs from each of the subsystems and processed it produce a final decision. This classifier system gave an identification success rate of 85.04%. Overlapping dataset was presented to the system and it gave better results when two vehicles are present in the same data window. The system was able to pick the data strain that was visible by most of the sensors as opposed picking the higher magnitude strain during the identification process.

The research shows that a ANN vehicle classification system is capable of an identification accuracy of over 90% for an isolated dataset and an overlapping vehicle strain measurements derived from measured strain in a bridge structure.

5.1 FUTURE RECOMMENDATIONS

The research reported here focused on data generated from simple software models, the classifiers should therefore be used on real strain SHM data. It has been observed that data collected from the field is not necessarily very clean. It contains noise which is present due to sampling, vibrations and oscillations including the natural frequency of the bridge and numerous other factors. It would be appropriate to see how the system behaves with the noise present.

The noisy data can be cleaned using software filters although this can remove useful details present in the raw strain data. The behaviour of the network using clean data can be compared to the behaviour when noisy data is used.

Although this research focused on performing data analysis without the use of any other data segmentation/analysis techniques, additional signal processing could possibly further aid the recognition mechanism. Additionally location of the event and favourably placing it in an observation window would further aid the system analyser.

6 REFERENCES

1. S. C. Ahalt, A. K. Krishnamurthy, P. Chen, and D. E. Milton, "Vector Quantization using Frequency-Sensitive Competitive-Learning Neural Networks," IEEE International Conference on Systems Engineering, pp. 131–134, 1989.

2. S. C. Ashalt, A. K. Krishnamurthy, P. Chen, and D. E. Milton, "Competitive Learning Algorithms for Vector Quantization," Neural Networks, Vol. 3, pp. 277–290, 1990.

3. A. K. Krishnamurthy, S. C. Ahalt, D. E. Melton, and P. Chen, "Neural Netwroks for Vector Quantization of Speech and Images," IEEE journal on Selected Areas of Communication, Vol. 8, no. 8, pp. 1449-1457, 1990.

4. Matlab® from MathWorks, http://www.mathworks.com/products/matlab/

5. L. Card, "Unsupervised Neural Computation for Event Identification in Structural Health Monitoring Systems," M.Sc. thesis, Dept. of Electrical and Computer Eng., Univ. of Manitoba, 2004.

6. C. M. Bishop, "Neural Networks for Pattern Recognition," Oxford University Press, 1995.

7. C. Sun, "An Investigation in the Use of Inductive Loop Signatures for Vehicle Classification," Institute of Transportation Studies, California Partners for Advanced Transit and Highways (PATH), 2000.

8. Mariana Ruiz Villarreal, "Complete_neuron_cell_diagram", Wikipedia, The Free Encyclopedia. 7 Feb 2008, 08:08 UTC. Wikimedia Foundation, Inc, 8 Feb 2008 http://en.wikipedia.org/wiki/Image:Complete_neuron_cell_diagram.svg, 2007.

9. R. C. Tennyson, et. al., "Structural Health Monitoring of Innovative Bridges in Canada with Fiber Optic Sensors," Smart Materials and Structures, Vol. 10, pp. 560-571, 2001.

10. "Neuron," Wikipedia, The Free Encyclopedia. 7 Feb 2008, 08:08 UTC. Wikimedia Foundation, Inc. 8 Feb 2008 <http://en.wikipedia.org/w/index.php?title=Neuron&oldid=189691187>.

11. D. K. McNeill, "Adaptive Visual Representations For Autonomous Mobile Robots Using Competitive Learning Algorithms," Ph.D thesis, Dept. of Electrical and Computer Eng., Univ. of Manitoba, 1998.

12. S. J. Peters, "Hardware Implementation of Stochastic Neural Networks," M.Sc. thesis, Dept. of Electrical and Computer Eng., Univ. of Manitoba, 2004.

13. D. K. McNeill, and L. Card, "A mixture of experts approach for SHM measurement monitoring," Proceedings of SPIE, 2006.

14. L. Card, and D. K. McNeill, "Novel event identification for SHM systems using unsupervised neural computation," Proceedings of SPIE, 2004.

15. P. V. Balakrishnan et. Al., "A study of the classification capabilities of Neural Networks using unsupervised learning: A comparison with K-Means Clustering," Psychometrika, Vol. 59, No. 4, pg. 509-525.

16. D. K. McNeill, H. C. Card, and A. F. Murray, "Unsupervised Cluster Identification for Embedded Systems," Proceedings of the Second ICSC Symposium on Engineering of Intelligent Systems, Paisley, UK, 2000.

17. J. Hertz, and A. Krogh, R. Palmer, "Introduction to the Theory of Neural Computation," Redwood City, CA: Addison Wesley, 1991.

18. D. E. Rumelhart, and D. Zipser, "Feature Discovery by Competitive Learning," Cognitive Science, 9, pp. 75-112, 1985.

19. J. Moody, and C. J. Darken, "Fast Learning in Networks of Locally-Tuned Processing Units," Neural Computation, 1, pp.281-294, 1989.

20. D. DeSieno, "Adding a Conscience to Competitive Learning," IEEE International Conference on Neural Networks, San Diego, Vol. 1, pp. 117-124, 1988.

APPENDIX A

Red River - North Perimeter Bridge
Sensor Location Diagram