SP
COSC
THO
95/96
RESERVE

Continuous Adaptive Prediction of Highway Traffic Volumes using Artificial Neural Networks

David J. Thompson

Thesis Submitted to the Department of Mathematics and Computer Science
in partial fulfilment of the requirements for the
Degree of Specialized Bachelor of Science in Computer Science

Supervisor: Dr. Pawan J. Lingras

Laurentian University

Department of Mathematics and Computer Science
Faculty of Science
Laurentian University
April 1996

© David J. Thompson

Abstract

This report details the investigation into the effect of several *artificial neural network* (ANN) architectures on the ability of these networks to predict *highway traffic volumes* (HTVs). The scope of this project is intended as a basis for further research by determining favourable choices for an underlying network architecture which may be expanded to incorporate recurrent temporal sensitive features. This will be determined on the basis of the errors that the predicted traffic volumes deviate from the target values in the training set. The training phase of the models will exclude data for up to a year to allow for the performance evaluation of the trained networks.

Acknowledgments

Table of Contents

List of Illustrations

List of Tables

CHAPTER 1

INTRODUCTION

The knowledge of short term performance is useful in day-to-day decision making regarding scheduling and resource allocations. Modelling of short term performance will include the continuous and adaptive prediction of the immediate future demand for the facilities. Statisticians and computer scientists have used a variety of techniques such as different variations of autoregressive models, multivariate adaptive regression splines, and recurrent or time delay neural networks. Each of these methods has different advantages and disadvantages. This project shall test models for highway traffic analysis by exploring the use of neural networks. It will begin with non-recurrent neural network designs which may then progressively combine the useful features of the existing techniques to develop a spatio-temporal network model for prediction of highway traffic usage on a continuous basis.

1.1 Scope of Project:

The purpose of this project has been modified from the original goals outlined above due to time constraints. The new objective is to test several different designs for the proposed neural networks which may later be further developed to accommodate the original goals of creating temporal recurrent network models. The different designs developed and tested under this project shall provide an indication of which general structures will be best suited for continuous prediction of *highway traffic volumes* (HTVs).

All the models will be tested for highway traffic based on existing data but may be extended to model computer network usage. The models developed for computer network usage may be used to evaluate alternate schedules and may even be extended to include performance analysis of other computer facilities. The results of this project may be compared with traditional statistical methods to determine the relative effectiveness of each as a predictive model or to propose a potential merging of neural network and statistical methodologies.

This project will consider two different data models with variations on the network architectures and the predicted output sets. The use of different input combinations shall be explored by using the traffic volumes from the previous week of the current year, and those from the current week of the previous year as input. Subdividing the problem domain shall also be tested in two ways. The two different input combinations shall be used on an hourly basis to predict each of 168 hourly traffic volumes for a given week, and again on a daily basis to predict the seven daily traffic volumes for a given week. Each of these shall incorporate two different choices for the number of hidden layer neurons to use, which generates eight different ANNs so far. The second subdivision scheme, predicting daily volumes, shall be repeated using seven individual networks, one for each day of the week. The seven individual daily networks may then be recombined in a "wrapper" network, thus becoming indistinguishable from a single larger unit to the external environment. They shall be tested with only the previous week of the current year as input.

The design schemes above shall result in fifteen networks in total for training. This should be adequate to provide a good indication of the best approach to use as the foundation for future modifications to incorporate time-delay constructs. The scope of this work and the time constraints of an under-graduate thesis project prevent the pursuit of the original goals presented in the original proposal. Namely, the networks shall be designed, constructed, and trained but, not tested. However, they will be trained with a subset of the available data to permit testing at a later date with up to a full year of previously unused data.

The measure of effectiveness or predictive correctness of the networks shall be made with a basic comparison of the percentage errors between the actual target values and the network estimated values. Unfortunately, the comparison of the neural networks' predictive ability with that of more traditional statistical methods will not be addressed at this time.

1.2 ANN Concepts:

ANNs are greatly simplified general models based on the structure and behaviour of the human brain which are composed of nodes, equivalent to biological neurons, and connections, analogous to biological synapses. Neural networks are graph structures which are generally fully connected between layers initially, then the training will adjust the connection weights and those which may not be required for, or influence, the output will be reduced, possibly to zero (i.e. eliminated). The ANN life cycle consists of the following stages: a construction phase, a training phase, a testing phase, and a implementation phase.

These phases of the neural network life cycle occur in the order listed above with the initial construction phase having the obviously intuitive meaning. The training phase is an adaptive, or dynamic, process when the ANN learns how to process the data based on a series of known data presented as input which are also used to compare with the estimated output. The estimated output values and actual target output value provide the means to calculate the error by which the estimated values deviate from the targets. This error is used during back-propagation to modify each connection weight that influences a particular output so that it would have generated the correct output during this cycle. This is repeated for an entire epoch consisting of a predetermined number of cycles (25000 maximum in this case) each consisting of the entire data set (approximately 250 weeks of data covering five years in this case).

The testing phase is a static process where the trained network is employed to predict outputs based on a previously unused known data set. Any errors generated during this phase are recorded to measure the network performance but, not used to modify the structure of the network. If the errors fall below a predefined threshold the network may be implemented, otherwise a determination must be made on where the problem lies and how to correct it. This would repeatedly generate a new network structure which may be trained on the same training data set previously employed until satisfactory results are obtained.

The implementation phase also has an obviously intuitive meaning, however there are two major methods which may be used at this point. A common way of implementing the trained and tested ANN is in a static state which is adequate if the problem domain presents a reasonably stable situation which the ANN is to model. However, some systems, such as the highway traffic prediction model, present an ever changing state (always increasing in this case) which would render a static model ineffective after a period of time. This is where continuous adaptive prediction would be of benefit such that the model should be non-linear and have the ability to predict future demand on a continuous basis. Using a continuous adaptive technique the actual values of the predicted demand will be used to adapt the network to improve its prediction ability throughout its useful lifetime.

The remainder of this report is organized in the following manner:

Chapter 1 presents the background information required for understanding the content of this report and the context of the material within the stated problem domain.

Chapter 2 presents the material reviewed in the process of the research performed to formulate this discussion.

Chapter 3 presents the methods and materials employed in this project, and the problems encountered.

Chapter 4 presents the results obtained, along with some statistics regarding the performance of the networks and the resources used to create them.

Chapter 5 presents derived conclusions and solutions for overcoming the problems encountered. Also included here are recommendations for potential further analysis or research.

CHAPTER 2

BACKGROUND

At this point an overview of some of the general information discovered during the literature review required for this undertaking shall be presented. This shall begin with some general definitions of terms relevant to Artificial Neural Networks then a discussion of the Ulbricht experiments.

2.1 ANN Terminology:

Node

This is the artificial equivalent of a biological neuron. The nodes are grouped into three classes of layers, Input, Hidden, and Output. This naming convention simply refers to the logical arrangement of the nodes from an external black-box perspective such that input nodes can accept data from outside world, output nodes can send data to outside world, and hidden nodes cannot be referenced externally at all.

Bias node:

Is generally set to a value of one and used as an additional input for all first hidden level nodes to act as a damper which prevents erratic data from adversely changing the weights prematurely. Also, it will aid in slowing down the rate that the connection weights change, which will prevent excessive oscillations on those changes.

Connection

This is the artificial equivalent of a biological synapse. The connections are weighted with a value used to determine how much significance each individual node will contribute to the summed input of the node that it is connected to.

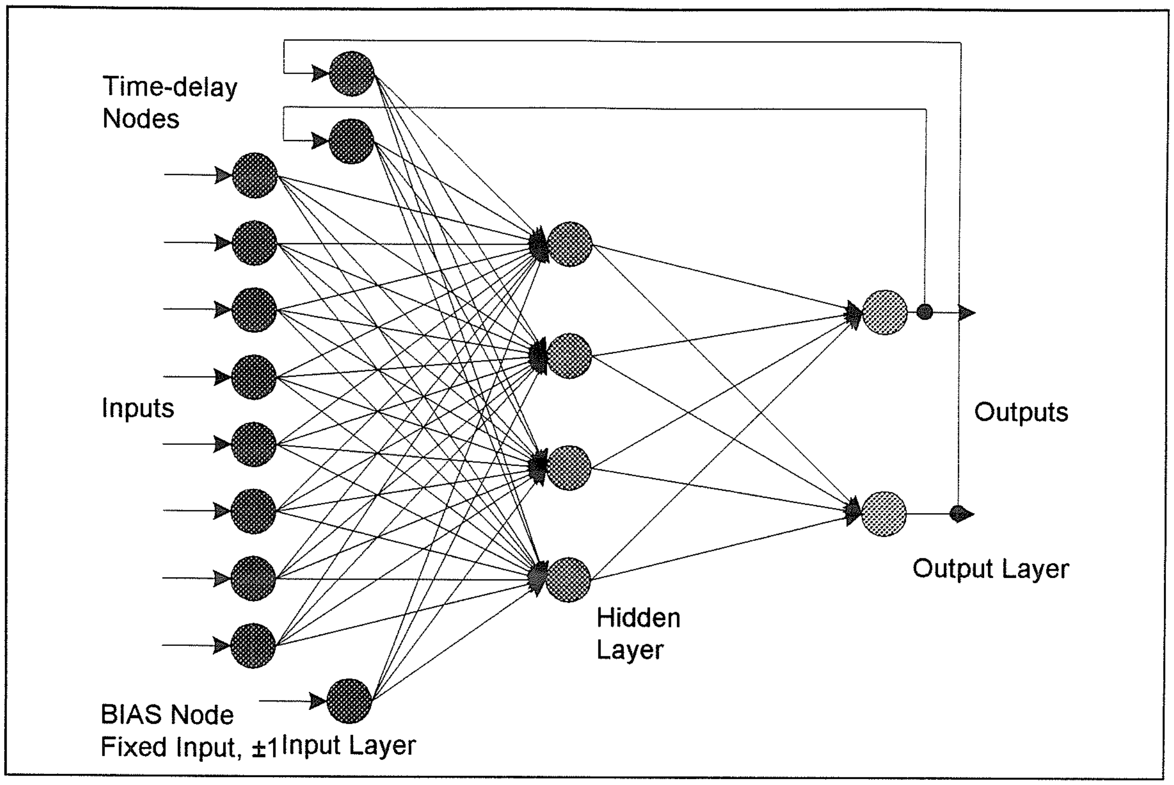

Figure 2.1 Nonlinear Model of an Artificial Neuron

Connection Weight

This is usually set to a value between -1 and +1. A positive weight enhances the influence of input on a result, and a negative weight inhibits inputs which are not required or less important to the result. They are not necessarily between 0 and 1 however, normalizing the values to between 0 and 1 prevent the ANN from producing excessively large output values, also this will prevent runaway accumulation of trends in data.

Epoch

One complete propagation of all sets of input values through the entire network. This includes cycles through the network which propagate individual data sets in a feedforward and back-propagation direction. There is a great deal of conflict in the literature on this terminology so this report will use the convention that an epoch indicates the use of the entire training set.

Transfer (Activation) Function

The mathematical equation employed to derive the value applied to the node input from the summed outputs of the preceding level. The three basic types are: Threshold (step) Function, Piece-wise Linear Function, and Sigmoid Function. The threshold values on the linear functions are arbitrary, and dependant on your application. The sigmoid family of functions are the most commonly used. They can be any nonlinear function which is smooth and continuous, thus, differentiable. This is because the inverse of the transfer functions are used as back-propagation (error) functions which are used during the training phase of the ANN life cycle

Error (Back-Propagation) Function

The mathematical equation employed to derive a "corrected" weight based on its resultant output error. If the transfer function was non-linear (i.e., a sigmoid function), this will be the differential of the transfer function. In that case, a transfer function is generally chosen which will be easy to differentiate.

Feedforward

Feedforward simply refers to the flow of data through the network during each cycle (epoch). This applies to the training, testing, and implementation phases of an ANNs life span.

Back-propagation

Refers to the adjustment of connection (synapse) weights based on output errors (i.e., the difference between the predicted and target output values) which are applied from output to hidden, and hidden to input layers successively. This applies to the training and adaptive implementation phases.

Recurrent (Feedback-propagation)

Refers to the method where connections are added which use the output of one or more neurons as input applied to other neurons in the network during the successive network cycle. These additional connections apply their inputs to the given neurons summed along with the original input values from all existing feedforward connections. Recurrent

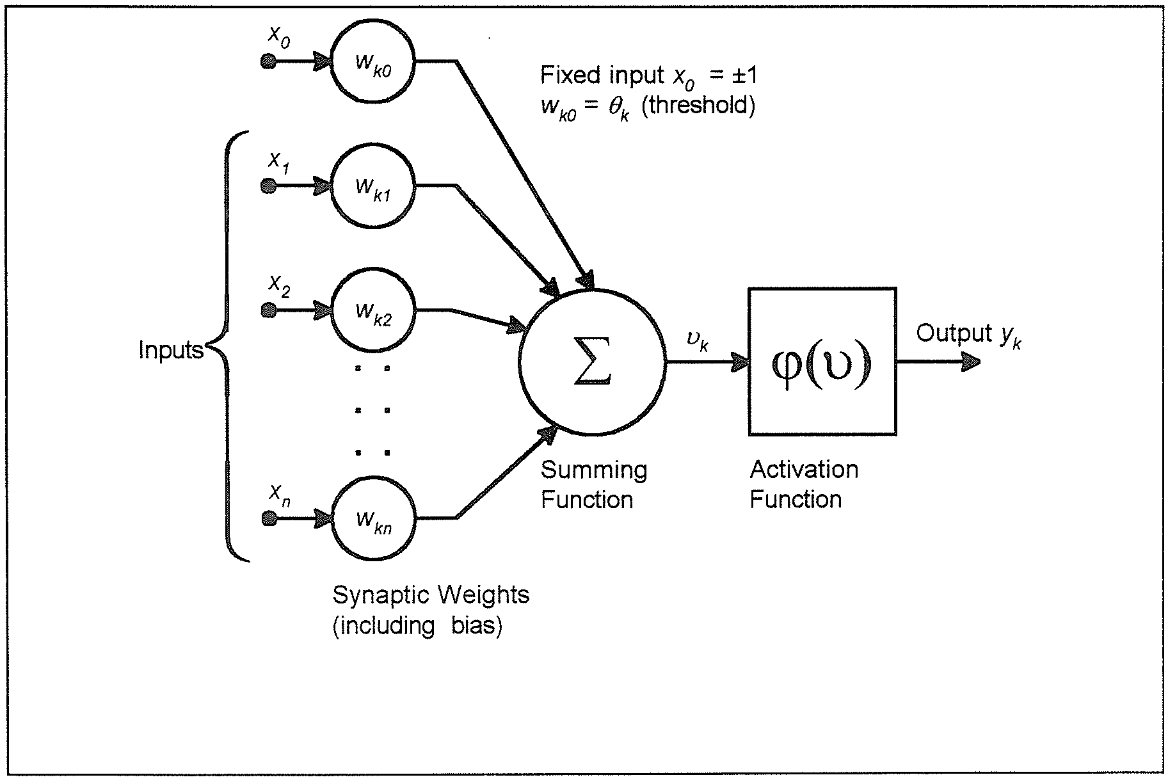

Figure 2.2 Fully Connected Recurrent Artificial Neural Network

networks are a special case of time-delay ANNs where the delay is one cycle rather than two, three, or more. This method applies to the training, testing, and implementation phases, however, this is a network structural concept as well as a data flow issue like the previous two definitions.

2.2 Literature Review:

This section is primarily based on the papers of Dr. Pawan Lingras and Claudia Ulbricht listed in the references and includes some of the relevant information derived from them.

This project is loosely based on the publication: Multi-recurrent Networks for Traffic Forecasting by Claudia Ulbricht at the Austrian Research Institute for Artificial Intelligence. The methods described in the Ulbricht experiments included: Windowed Network, Hidden Layer Feedback, Output Feedback, and Input Memory techniques which were a progression from basic feedforward networks to multi-recurrent networks. In all cases the networks consisted of the basic input-hidden-output node combination of 36-10-1, respectively.

Ulbricht Experiments

The methods described in the Ulbricht experiments are:

Windowed Network

This is a simple feedforward network

"Windowed" refers to how the time-frame of the selected input advances throughout the life of the network

i.e., as we advance in time the input window (one week of data in this case) will also advance, so that the relative temporal distance between the week being predicted and the data used as a basis for that prediction remains constant

This corresponds to the input data window "jumping" forward a week at a time, there is no overlapping of hours from one week to the next

Hidden Layer Feedback

This involves the addition of feedback of the hidden layer outputs to an additional layer of nodes (called the Context layer) which is then summed with the outputs of the input layer on the next updating phase (epoch) as the new input to the hidden layer

This was modified later to implement a "memory" layer with self-feedback loops to itself (as explained below) by duplicating the Context layer three times with different weights applied to their inputs and self-feedback loops

Output Feedback

Same feedback concept as above applied to the output layer using its own Context layer

Feedback values in this case were composed of the network outputs, the target outputs, and the differences between the two

This was also modified as stated in the previous method (explained below)

Input Memory

Same feedback concept as above applied to the input layer using its own Context layer

This was also modified as stated in the previous method (explained below)

All of these feedback methods were combined to produce the final ANN of this experiment which has twelve Context layers performing feedback to emulate both short- and long-term memory

Memory Layer

Each of the preceding feedback links were modified to provide a persistent "memory" as follows:

Each initial Context layer accepted input values weighted 100% and output them with no further self-feedback

This effectively allows the network to "remember" the previous input which is then combined with the current inputs to the hidden layer

ALL Context layers output to the same hidden layer of neurons

The Context layer of each network level was duplicated three times with various weights as follows:

Original	100% input	0% self-feedback
Duplicate 1	75% input	25% self-feedback
Duplicate 2	50% input	50% self-feedback
Duplicate 3	25% input	75% self-feedback

This creates "memory cells" of different flexibility such that:

A smaller weight applied to the input to a Context layer generates a memory cell which is less influenced by recent changes of input

A smaller weight on the self-feedback loop of a Context layer generates a memory cell which is a more flexible short-term memory

Thus, we have a progression from flexible short-term memory cells (Original Context layer) to more rigid long-term memory cells (25% input, 75% self-feedback)

CHAPTER 3

THE EXPERIMENTS

3.1 Methods:

- state what I did

- list material and equipment

The ANNs were constructed using the Stuttgart Neural Network Simulator (SNNS) Version 4.0 software running on a Sun SuperSparc 20, 50 MHz computer with 64 MB of RAM. Initially eight different non-recurrent ANNs were created in hopes of learning which of these would provide the best results and expand on these in the next phase of Dr. Lingras' continuing research. The selected ANN architectures will be refined to create recurrent ANNs, then possibly time-delay recurrent ANNs.

3.1.1 Initial Non-recurrent AANs:

Two distinct models were chosen as a starting point for the experiments where the difference was in predicting either the hourly traffic volumes for each of the 168 hours in a week, or the daily traffic volumes for each of the seven days in a week. This was further enhanced by using two variances on the time window chosen as input for the week to be predicted. These two schemes were to use either the traffic volumes from the previous week of same year as input, or the traffic volumes from the same week of previous year as input.

The general structure of these ANNs were defined as presented in Table 1 below. This selection of the number of nodes to use in each layer was determined in the following manner.

Table 3.1 Basic Structure of Initial ANNs Based on Number of Neurons Used

	Input	Hidden	Output
Net 1	186	93	168
Net 2	186	177	168
Net 3	25	13	7
Net 4	25	16	7

3.1.2 Input Nodes:

The number of input nodes was selected as 186 = 168 + 12 + 5 + 1 for the hourly model, and as 25 = 7 + 12 + 5 + 1 for the daily model. These figures correspond to 168 HTVs of each hour of week, or seven HTVs of each day of week, plus twelve flags representing the month of year, plus five flags representing the week of month, plus one flag representing the occurrence of a holiday on the week in question. An input node value can represent a range of values a or a flag (-1, 0, or +1 in this case) indicating the presence or absence of a particular condition. A flag is useful for values which should not be interpreted as scalars.

A flag is useful for values which should not be interpreted as scalars

ie. the use of one node with a range of 1 - 12 indicating the month may train the network to interpret Dec. (12) as somehow better than Sept. (9) since it is numerically larger

The choice of flags were suggested as meaningful in the context of highway traffic patterns where it should be reasonable to expect fluctuations in traffic volumes based on details such as the day of the week, whether or not this is a holiday week, and the like. Unique nodes representing flags to indicate the month or week in question are deemed as the best method because a single node with multiple values will train the ANN such that it will interpret Dec. (12) as "better than" Sept. (9) since it is numerically larger. In some applications that may be desirable but, here it is not.

3.1.3 Output Nodes:

The number of output nodes is simply a matter of the number of HTVs to be predicted. That is, 168 is simply the number of hours in one week, and seven is simply the number of days in one week to be predicted.

3.1.4 Hidden Nodes:

The number of hidden nodes were based on two rules-of-thumb generally used for this selection. The two methods were, the number of input nodes divided by two, or the sum of the input and output nodes divided by two. In both cases the result was rounded up to the nearest integer value. Thus, the first method generated 93 and thirteen hidden nodes, and the second method generated 177 and sixteen hidden nodes.

The first method for determining the number of hidden nodes to use is based on a theorem (?) which shows that this all that is required to model any continuous function. In general, the design of neural networks may be considered as much an art as a science and it is considered prudent to "start with small nets and then add more and more state layers later, try[ing] to keep the network as small as possible." (Ulbricht, e-mail)

- describe research methods and technical procedures followed

3.2 SNNS Software:

The *Stuttgart Neural Network Simulator* (SNNS) package proved to be a very complete and effective package, it is provided free of charge by anonymous ftp from the University of Stuttgart at the internet address provided in the references (Zell).

3.2.1 Creating the Networks:

Creating the networks was a very simple process, beginning by starting the simulator with the command "ssns" which then displays the copyright banner that must be clicked on to proceed. A single click on a button labelled "BIGNET" allows you to choose the type of network desired, for this experiment the feedforward type was selected. For each layer desired selections were made, in order from the input, to the hidden, then the output layer, as indicated in the following table, each followed by clicking on the button labelled "insert".

Table 3.2 Network Configuration Options

layer	type:	#units in x-dir	#units in y-dir:	#units in z-dir:	relative pos'n:
input	input	186	1	1	below
hidden	hidden	177	1	1	below
output	output	168	1	1	below

Next the option was selected for "full connection", then "create net", then "done". From this point the file creation was performed by selecting the button labelled "FILE", then the option for file type "net", and the network name "186-177-168" based on the example in the table above.

This naming convention was adopted throughout to aid in identifying the networks based on their physical structure via the number of nodes used in the respective layers. At this point the option "save" was selected, the sub-menu exited by selecting "done". When all the different network architectures were created in this fashion the button labelled "QUIT", then the option "yes" were selected to end the process. All very simple to execute with no real understanding of the underlying processes involved required of the user.

3.2.2 Creating the Input Patterns:

The Stuttgart Neural Network Simulator software suite requires what is known as pattern files for input to the actual network. These pattern files are ASCII text files consisting of the repeated sequence of a line representing the network input followed by the

CHAPTER 4

RESULTS

The following data results are the errors of the predicted HTVs which are the differences between the estimated volumes and the target volumes calculated as percentages of the actual target HTVs. Also included here are statistics generated by the SNNS package which indicate the hardware resources required to train the networks. The creation of the networks, as described in the previous chapter, was a relatively trivial matter due to the very high quality of the SNNS software package.

4.1 Estimation Errors:

The data results presented here have a minor flaw that was detected rather late in the process. An oversight was made regarding the condition where output values may take either positive or negative values, thus the maximum and minimum errors presented here do not fully describe the situation. The data will be calculated again with absolute values for all percentage errors to provide more significant results. However, the current results do partially confirm the proposed concepts regarding which ANN structures may best fit the given problem domain and are presented in the following tables.

Table 4.1 Results of Network: 186-177-168 Current Year

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
average	0.0725	0.0095	0.1661	-1.1113	0.0692	0.1152	0.1994
max	1.0000	0.9994	0.9952	0.9983	0.9949	0.9920	0.9996
min	-20.0000	-14.5491	-9.5159	-617.2500	-18.8525	-10.0077	-9.5297
median	0.2693	0.2167	0.3878	0.3829	0.2682	0.2549	0.3699

Table 4.2 Results of Network: 186-177-168 Previous Year

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
average	-0.0214	0.0400	0.1407	-0.6041	0.0991	0.1281	0.1878
max	0.9994	0.9994	0.9962	0.9973	0.9901	0.9920	0.9998
min	-86.5862	-37.4203	-12.2519	-607.2500	-17.0244	-12.4184	-9.4000
median	0.2895	0.2219	0.4238	0.4073	0.2872	0.2728	0.3942

Table 4.3 Results of Network: 186-93-168 Current Year

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
average	-0.1892	-0.2533	-0.0851	-1.0518	-0.2905	-0.1115	-0.0978
max	0.9497	0.9586	0.9595	0.9601	0.9605	0.8660	0.9353
min	-15.0000	-16.5019	-9.0973	-621.5000	-27.7612	-10.0533	-14.8041
median	0.1336	0.0341	0.1710	0.2428	0.1023	0.0717	0.1883

Table 4.4 Results of Network: 186-93-168 Previous Year

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
average	-0.4733	-0.0753	-0.1059	-0.9756	-0.2393	-0.0951	-0.0341
max	0.9440	0.9577	0.9757	0.9788	0.9710	0.8615	0.9616
min	-355.3571	-16.6207	-9.9520	-624.1250	-28.1343	-12.2444	-17.8814
median	0.0855	0.1125	0.1818	0.2255	0.1642	0.0920	0.2132

Table 4.5 Results of Network: 25-16-7 Current Year

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
average	-0.0179	0.0061	-0.0166	-0.2458	0.0040	0.0002	0.0047
max	0.2645	0.2073	0.2313	0.2191	0.2327	0.1743	0.2490
min	-6.6058	-0.3513	-5.1331	-65.2051	-0.7969	-0.5832	-0.5414
median	0.0102	0.0124	0.0057	0.0090	0.0061	0.0086	0.0123

Table 4.6 Results of Network: 25-16-7 Previous Year

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
average	-0.0153	-0.0108	0.0116	-0.2582	-0.0189	0.0002	-0.0120
max	0.1121	0.1564	0.1191	0.4119	0.2210	0.0939	0.1270
min	-0.9900	-0.2546	-0.1393	-40.2598	-0.3798	-0.1550	-0.1662
median	-0.0148	-0.0128	0.0161	0.0176	-0.0234	0.0069	-0.0168

Table 4.7 Results of Network: 25-13-7 Current Year

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
average	-0.0243	-0.0016	-0.0098	-0.2490	0.0036	-0.0047	-0.0007
max	0.2665	0.1921	0.2308	0.2121	0.2273	0.1575	0.1984
min	-6.4668	-0.3468	-5.0807	-64.7248	-0.7954	-0.6015	-0.6668
median	0.0081	0.0060	0.0048	0.0088	0.0117	0.0007	0.0127

Table 4.8 Results of Network: 25-13-7 Previous Year

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
average	-0.0142	-0.0110	-0.0066	-0.2668	0.0023	0.0036	-0.0057
max	0.1185	0.1598	0.1112	0.4047	0.2498	0.0903	0.1263
min	-0.9914	-0.2465	-0.1411	-43.3402	-0.3724	-0.1534	-0.1672
median	-0.0106	-0.0201	-0.0104	0.0191	0.0147	0.0113	-0.0049

Table 4.9 Results of Seven Individual Daily Volume Networks: 25-13-1 Current Year

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
average	-0.0228	-0.0036	-0.0116	-0.2581	0.0040	0.0060	0.0044
max	0.2680	0.1916	0.2375	0.2107	0.2229	0.1723	0.2526
min	-6.4404	-0.3020	-5.0636	-64.6974	-0.8217	-0.5830	-0.5220
median	0.0091	-0.0051	0.0068	-0.0021	0.0180	0.0173	0.0157

4.2 Processing Requirements:

The following tables illustrate some of the basic characteristics of each network. The tables are organized according to the commonalities present in each of the main structures used in this project. The abreviation CUPS refers to the number of *connection updates per second.* The main points of interest are the processor times required to train each network which effectively demonstrate the dramatic increase in overhead with larger networks. Two values of significance in the number of learned cycles category from table 4.10 are explained below.

Table 4.10 Performance Statistics: 168 Hourly Output Networks

Network name	186-177-168	186-177-168p	186-93-168	186-93-168p
No. of units	531	531	447	447
No. of links	62658	62658	32922	32922
No. of patterns	252	153	252	153
No. of cycles	25000	25000	25000	25000
Max. error to stop	0.001000	0.001000	0.001000	0.001000
No. of learned cycles	25000	25000	9216	3644
CPU time (sec)	391099.31	259815.09	66688.22	18781.01
CPU time (hr)	108.64	72.17	18.52	5.22
User time (sec)	465408	290185	154739	43179
User time (hr)	129.28	80.61	42.98	11.99
No. of CUPS	1.009323e+06	9.224516e+05	1.146516e+06	9.773206e+05

Table 4.11 Performance Statistics: Seven Daily Output Networks

Network name	25-16-7	25-16-7p	25-13-7	25-13-7p
No. of units	48	48	45	45
No. of links	512	512	416	416
No. of patterns	252	153	252	153
No. of cycles	25000	25000	25000	25000
Max. error to stop	0.001000	0.001000	0.001000	0.001000
No. of learned cycles	25000	25000	25000	25000
CPU time (sec)	2373.69	1824.96	2378.73	1289.06
CPU time (hr)	0.66	0.51	0.66	0.36
User time (sec)	12256	10219	12235	7007
User time (hr)	3.40	2.84	3.40	1.95
No. of CUPS	1.358897e+06	1.073119e+06	1.101764e+06	1.234388e+06

Table 4.12 Performance Statistics: One Daily Output Networks

Network name	25-13-1.1	25-13-1.2	25-13-1.3	25-13-1.4
No. of units	39	39	39	39
No. of links	338	338	338	338
No. of patterns	257	257	257	257
No. of cycles	25000	25000	25000	25000
Max. error to stop	0.001000	0.001000	0.001000	0.001000
No. of learned cycles	25000	25000	25000	25000
CPU time (sec)	1746.46	1745.65	1919.96	1743.27
CPU time (hr)	0.49	0.48	0.53	0.48
User time (sec)	12458	12170	12449	11824
User time (hr)	3.46	3.38	3.46	3.28
No. of CUPS	1.243458e+06	1.244035e+06	1.131091e+06	1.245734e+06

Table 4.13 Performance Statistics: One Daily Output Networks (cont.)

Network name	25-13-1.5	25-13-1.6	25-13-1.7	
No. of units	39	39	39	
No. of links	338	338	338	
No. of patterns	257	257	257	
No. of cycles	25000	25000	25000	
Max. error to stop	0.001000	0.001000	0.001000	
No. of learned cycles	25000	25000	25000	
CPU time (sec)	1745.66	1748.80	1756.36	
CPU time (hr)	0.48	0.49	0.49	
User time (sec)	10974	10960	10978	
User time (hr)	3.05	3.04	3.05	
No. of CUPS	1.244028e+06	1.241794e+06	1.236449e+06	

4.3 Discussion:

The anticipated outcome was that smaller ANNs would be more effective and that more accurate results would be obtained from a higher degree of specialization i.e., the smaller networks tailored to predict individual days of the week in this case. Although the results may not clearly demonstrate that this is the case, it is felt that recalculating them with absolute values of the percentage errors will reveal this to be true.

One exceptional characteristic of note in table 4.10 are the number of learned cycles entries for networks 186-93-168 using data from both the current year and the previous year. this indicates that the networks were trained to below the acceptable error levels very much earlier than all the other networks. This would indicate that these two networks are the best candidates for further study since they seem to have adapted to the problem set more readily than the other configurations. At the very least they should be the first to be statically tested with fresh data.

CHAPTER 5

INTERPRETATIONS

This discussion assumes that neural networks are at least as effective, if not more so, than statistical methods, and more tolerant of noisy or incomplete data (Lingras). Thus the main focus here has been to compare different ANN architectures and identify those more effective at predicting correct results in the chosen problem domain, namely analyzing HTVs. It must be noted that for all the network architectures created during the course of this project the data has not been subject to rigorous statistical analysis at this point. This project was merely intended to provide a starting point to explore the use of fully recurrent temporal ANNs.

5.1 Conclusions:

Among the conclusions drawn from performing this experiment are the following:

- The most notable finding is the very rapid rate at which the two networks predicting 168 hourly HTVs with the smaller number of hidden neurons (93) achieved the designated acceptable error level. This general configuation definitely warrants further consideration.
- The large networks providing 168 hourly HTVs per week seem to be marginally more effective with input data from the current year whereas, the smaller networks predicting seven daily HTVs are more effective with input data from the previous year.
- In all cases the fewer hidden nodes selected as a starting point is the best choice. More can be added later if results are not satisfactory.
- With such an unlimited number of possible configurations that may be used some form

of reasonable convention must be chosen as a guide in further study. Also, smaller networks may be processed in a shorter period of time, allowing for a broader spectrum of possibilities to be explored.

- Predictions for all days of the week behaved in a relatively uniform manner with the exception of Wednesdays which developed larger errors in all networks tested.
- Although it would be prudent to compare the results of the performance of the ANNs with statistical methods, it may not be absolutely necessary. This facet of the neural network methodology has been studied extensively and the general consensus is that ANNs are more effective and more flexible as predictive tools.
- There may be a variety of temporal, spatial, or local cultural factors which are unknown to this researcher at this point in time that may alter the results by their exclusion. It is known that the data collection site is very close to a large urban centre (less than 20 km north of Calgary) at a truck weigh station (Balzac scales). A better familiarity with the shipping routes and schedules for this part of the continent may provide more insight.

5.2 Recommendations:

In concurrence with advise provided by both Dr. Lingras and Ms. Ulbricht, it seems more appropriate to focus on smaller specialized neural networks. A few recommendations on how this work may be continued are:

- Testing of the network configuration noted above (186-93-168) should be performed with fresh data to confirm whether or not they sustain an acceptable degree of accuracy. They are the best candidates for further study, pending completion of the remaining calculations using absolute error values from all ANN results data.

- The spurious results for predictions of HTVs on Wednesdays, where the errors were much larger than those for any other day of the week, are due to an unknown cause at this point. It should be determined if there is a local condition or some other phenomena responsible for this which has been unanticipated as being relevant at present.
- The original inspiration for this project is still a viable extension of this work namely, to develop a continuous and adaptive prediction tool to model immediate future resource demand as a temporal model based on the "Input Memory" techniques described in the Ulbricht paper.
- The choice of creating an individual network for each day of the week seems the best approach. These may be combined if desired into a larger ANN structure to encompass an entire week of input. At present it is unclear to this researcher what the effectiveness of predicting an entire week of input may be, again further study may reveal this need.
- Another combination that may provide more insight is to use individual daily networks using the hourly HTVs in combinations of input-hidden-output neurons such as: 25-hh-1, 25-hh-24, 186-hh-1, or 186-hh-24. Where: hh indicates any appropriate number of hidden nodes, and output of 1 indicates a single daily HTV or an output of 24 indicates all the hourly HTVs for a given day.
- It is recommended that further modifications to encompass recurrent feedback techniques be based on some form of the individual daily networks. This should follow some study of the other combinations as outlined above.
- It may be wise to follow through on a comparison the results with same HTV system

modeled using traditional statistical methods at some future point. It would be more meaningful to do so with a more complete set of fully recurrent ANNs to provide confirmation of the prediction ability of artificial neural networks to compete with traditional statistical methods within the context of the highway traffic volume problem.

Bibliography

NeuralWare: An Overview of Neural Computing. Vers. 2.0. Diskette. Pittsburgh, PA: NeuralWare Inc., 1992.

Allman, William F. Apprentices of Wonder: Inside the Neural Network Revolution. NY, NY: Bantam Books, 1989.

Anderson, James A. An Introduction to Neural Networks. Cambridge, MA: The MIT Press, 1995.

Bharath, Ramachandran, and James Drosen. Neural Network Computing. NY, NY: Windcrest / McGraw-Hill, 1994.

Haykin, Simon. Neural Networks: A Comprehensive Foundation. NY, NY: Macmillan College Pub. Co., 1994.

Lawrence, Jeanette. Introduction to Neural Networks: Design, Theory, and Applications. 5th ed. Nevada City, CA: California Scientific Software, 1993.

Lingras, Pawan. “Role of Recurrent Backpropagation Neural Networks in Seasonal Data Collection Programs.” Sault Ste. Marie, ON: 12 pp.

Prechelt, Lutz. “neural-net-part1-7.faq.” ?? Sept. 1995. Online posting. Newsgroup comp.ai.neural-nets. Usenet. 18 Sept. 1995.

Ulbricht, Claudia. “Multi-recurrent Networks for Traffic Forecasting.” Vienna, Aus.: 11 pp.

Ulbricht, Claudia. “Re: Undergrad Thesis on Artificial Neural Networks.” E-mail to Dave Thompson. 17 Nov. 1995.

Zahedi, Fatemeh. Intelligent Systems for Business: Expert Systems with Neural Networks. Belmont, CA: Wadsworth Pub. Co., 1993.

Zell, Andreas, et al. Stuttgart Neural Network Simulator. Vers. 4.0. Online. Internet. Stuttgart, Ger.: U of Stuttgart, (IPVR), 1995. Available FTP: ftp.informatik.uni-stuttgart.de (129.69.211.2) /pub/SNNS/.

Appendix A

HOLIDAYS.DAT

Holidays.dat is a plain text file generated to identify which weeks of the year contained a holiday under the week numbering scheme adopted for this project. This scheme defines the first day of a week to be a Sunday and the last day of the week to be the following Saturday. Weeks are numbered within the month they occur where the first Sunday of the month defines the first week of that month. If a week begins in one month and ends in the next it is considered to belong to the first month. Each line in the file begins with an integer indicating the last two digits of a year, followed by integer pairs indicating the month of the year and week of the month, in that order, when a statutory holiday occurs.

A portion of holidays.dat:

```
Usage of holidays.dat: Throw away first 4 lines, format data as
2 digits for year, repeat 1 integer for month, 1 integer for week
that each holiday falls on. (all separated by one whitespace)


85 3 5 4 1 5 3 6 5 8 1 9 1 10 2 11 2 12 4
86 3 4 3 5 5 3 6 5 8 1 8 5 10 2 11 2 12 3
87 4 2 4 3 5 3 6 4 8 1 9 1 10 2 11 2 12 3
...
95 1 1 4 2 4 3 5 3 7 1 8 1 9 1 10 2 11 2 12 4
96 3 5 4 1 5 3 6 5 8 1 9 1 10 2 11 2 12 4
```

Appendix B

AN EXAMPLE SNNS CONFIGURATION FILE

```
Type: SNNSBATCH_2

NetworkFile: 186-177-168.net

#

InitFunction: Randomize_Weights

NoOfInitParam: 2

InitParam: -1.0 1.0

#

LearnPatternFile: c002181t-186-nn-168.pat

NoOfLearnParam: 2

LearnParam: 0.05 0.05

MaxLearnCycles: 25000

MaxErrorToStop: 0.001

Shuffle: YES

#

TrainedNetworkFile: trained_186-177-168.net

ResultFile: 186-177-168.res

ResultIncludeInput: NO

ResultIncludeOutput: YES

#
```