PERGAMON Mathematical and Computer Modelling 33 (2001) 683–693

MATHEMATICAL AND COMPUTER MODELLING

www.elsevier.nl/locate/mcm

An Application of Artificial Neural Networks for Rainfall Forecasting

KIN C. LUK, J. E. BALL AND A. SHARMA
Water Research Laboratory
School of Civil and Environmental Engineering
The University of New South Wales, Australia

Abstract—Rainfall forecasting is important for many catchment management applications, in particular for flood warning systems. The variability of rainfall in space and time, however, renders quantitative forecasting of rainfall extremely difficult. The depth of rainfall and its distribution in the temporal and spatial dimensions depends on many variables, such as pressure, temperature, and wind speed and direction. Due to the complexity of the atmospheric processes by which rainfall is generated and the lack of available data on the necessary temporal and spatial scales, it is not feasible generally to forecast rainfall using a physically based process model. Recent developments in artificial intelligence and, in particular, those techniques aimed at pattern recognition, however, provide an alternative approach for developing of a rainfall forecasting model. Artificial neural networks (ANNs), which perform a nonlinear mapping between inputs and outputs, are one such technique. Presented in this paper are the results of a study investigating the application of ANNs to forecast the spatial distribution of rainfall for an urban catchment. Three alternative types of ANNs, namely multilayer feedforward neural networks, partial recurrent neural networks, and time delay neural networks, were identified, developed and, as presented in this paper, found to provide reasonable predictions of the rainfall depth one time-step in advance. The data requirements for and the accuracy obtainable from these three alternative types of ANNs are discussed.

Keywords—Artificial neural networks, Comparison, Rainfall forecasting.

1. INTRODUCTION

A challenging task for catchment management and flood management in particular is the provision of a quantitative rainfall forecast. Accurate forecasts of the spatial and temporal distribution of rainfall are useful for both water quantity and quality management. For example, a flood warning system for fast responding catchments may require a quantitative rainfall forecast to increase the lead time for warning. Similarly, a rainfall forecast provides advance information for many water quality problems. The aim of this study is to provide short-term rainfall forecasts for multiple locations within an urban catchment.

There are two possible approaches to forecast rainfall. The first approach involves the study of the rainfall processes in order to model the underlying physical laws. However, this physically

The authors wish to acknowledge the Australian Research Council for financial support for the study. The authors also wish to acknowledge the Upper Parramatta River Catchment Trust for providing data for the study. The authors would like to express their thanks to the reviewer's constructive comments which have enhanced the readability of the paper.

0895-7177/01/$ - see front matter Typeset by AMS-TEX
PII: S0895-7177(00)00272-7

based process modelling approach may not be feasible because

- rainfall is an end product of a number of complex atmospheric processes which vary both in space and time;
- even if the rainfall processes can be described concisely and completely, the volume of calculations involved may be prohibitive; and
- the data that is available to assist in definition of control variables for the process models, such as rainfall intensity, wind speed, and evaporation, etc., are limited in both the spatial and temporal dimensions.

A second approach to forecast rainfall is based on the pattern recognition methodology, which attempts to recognise rainfall patterns, based on their features. The logic behind this approach is to find out relevant spatial and temporal features in historical rainfall patterns and to use these to predict the evolution of other storms. This pattern recognition technique makes reference only to rainfall patterns, with no consideration of the physics of the rainfall processes. This approach is considered appropriate for the present study because the main concern is to forecast short-duration rainfall at specific locations within a catchment. A thorough understanding of the physical laws usually is not required, and the data requirements are not as extensive as for a process model.

The rainfall forecasting models developed in this study are based on the use of ANNs to implement the pattern recognition methodology. ANNs, which emulate the parallel distributed processing of the human nervous system, have proven to be very powerful in dealing with complicated problems, such as pattern recognition and function approximation. It has been shown by Hornik *et al.* [1] that an ANN with sufficient complexity is capable of approximating any smooth function to any desired degree of accuracy. In addition, ANNs are computationally robust, in the sense that they have the ability to learn and generalise from examples to produce meaningful solutions to problems even when input data contain errors or are incomplete. A further advantage of ANNs in relation to short-term rainfall forecasting is that ANNs can be designed to operate in real-time.

There have been a number of reported studies that have used ANNs to solve problems in hydrology. For example, French *et al.* [2] used an ANN to forecast rainfall for a catchment with artificial rainfall inputs, while Hsu *et al.* [3] applied an ANN to model the rainfall-runoff process. Most previous studies have used a multilayer feedforward network (MLFN), without considering the other types of ANNs. Although the MLFN is a powerful nonlinear model, there are many other types of ANNs which may have been more appropriate to the problem. ANNs appropriate for rainfall forecasting include the partial recurrent neural networks (PRNN) and time-delay neural networks (TDNN).

Presented in this paper is a comparison of three types of ANNs (i.e., MLFN, PRNN, and TDNN) for forecasting rainfall over an urban catchment in western Sydney, Australia. The ANNs were applied to forecast rainfall at multiple locations simultaneously for the next 15 minutes based on previously observed spatial and temporal rainfall patterns. The important issues of identifying the order of lag and complexity of the networks are discussed also.

2. THE ARTIFICIAL NEURAL NETWORKS APPROACH

2.1. The Basics

An ANN is a computational approach inspired by studies of the brain and nervous systems in living organisms. It is believed that the powerful functionality of a biological neural system is attributed to the parallel distributed processing nature of a network of cells, known as neurons. An ANN emulates this structure by distributing the computation to small and simple processing units, called artificial neurons, or nodes. With this architecture, an ANN has proven to be a powerful mathematical model which excels at function approximation and pattern recognition.

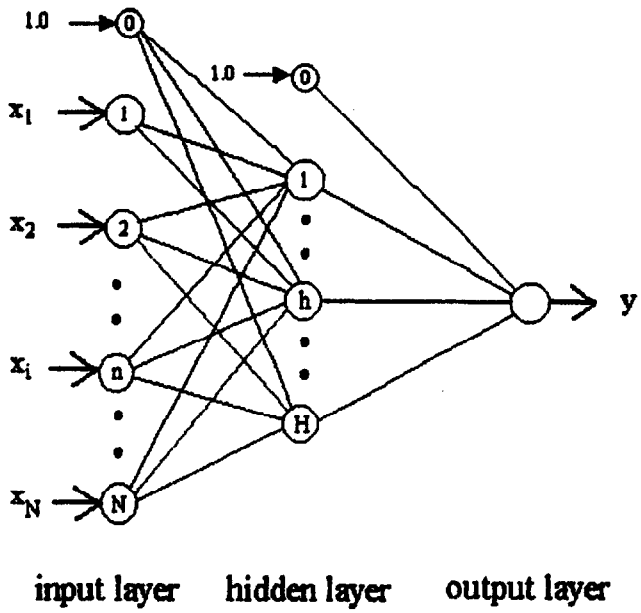

Figure 1. A simple three-layer feedforward neural network.

An ANN is formed by nodes connected together. Nodes with similar characteristics are arranged into a layer. A layer can be seen as a group of nodes which have connections to other layers or the external environment, but which have no interconnections. Shown in Figure 1 is a simple ANN which consists of three layers of nodes.

There are basically three types of layers. The first layer connecting to the input variables is called the input layer. The last layer connecting to the output variables is called the output layer. A single output node is shown in Figure 1 for clarity of illustration. It is straightforward to extend to multiple output nodes. Layers between the input and output layers are called hidden layers; there can be more than one hidden layer. Information is transmitted through the connections between nodes. In a simple situation, information is passed forward only, as shown in Figure 1. This type of network is called a feedforward network, or multilayer feedforward network (MLFN).

Mathematically, a three-layer MLFN with N input nodes, H hidden nodes, and one output node, can be expressed in the following formula:

$$y = S_1\left(\sum_{h=1}^{H} O_h \cdot w_h + w_0\right), \tag{1}$$

where

y is the output from the ANN,

O_h is the output value of the h^{th} hidden node

$$O_h = S_2\left(\sum_{n=1}^{N} x_n \cdot w_{nh} + w_{0h}\right), \tag{2}$$

x_n are the inputs to the MLFN,

w_h are the connection weights between nodes of the hidden and output layer,

w_{nh} are the connection weights between nodes of the hidden and the input layer,

$x_o = 1.0$ is a bias and w_o and w_{oh} are the weights for the biases (the biases are used to prevent the error surface from permanently passing through the origin), and

s_1 and s_2 are activation functions. The most commonly used activation function is a logistic sigmoid function

$$s(x) = \frac{1}{1+e^x}. \tag{3}$$

The main parameters of the MLFN are the connection weights. The estimation of parameters is known as "training" in which optimal connection weights are determined by minimising an objective function.

2.2. ANN for Rainfall Forecasting

The scope of the present study is to forecast rainfall one time-step ahead. For the development of the proposed rainfall forecasting models, the rainfall process was assumed to be a Markovian process, which means that the rainfall value at a given location in space and time is a function of a finite set of previous realisations. With this assumption, a model structure can be expressed as

$$X(t+1) = g(X(t), X(t-1), X(t-2, \ldots, X(t-k+1)) + e(t), \tag{4}$$

where

$X(t)$ represents a vector of rainfall values $x_{1t}, x_{2t}, \ldots, x_{Nt}$ at N different locations at time t,
$g(\,)$ is a nonlinear mapping function, which will be approximated using an ANN,
$e(t)$ is a mapping error (to be minimised), and
k is the (unknown) number of past rainfall realisations contributing to rainfall at the next time-step; usually, k refers to the lag of the network; if $k = 1$, the rainfall at the next time-step is related only to the present rainfall, thus giving a lag-1 network.

The development of an ANN for rainfall forecasting involves the following important considerations:

- select an appropriate ANN to represent the Markovian process;
- estimate the lag for the ANN, i.e., to determine the number of past rainfall values to be included as inputs;
- determine the optimal complexity of the ANN appropriate to the problem, i.e., to determine the number of hidden layers, and number of nodes in a hidden layer.

2.3. Alternative Networks

There are a number of alternative ANNs which can be used to represent the Markovian model expressed in equation (4), where the continuous process of rainfall is being represented by a discrete process. Based on a review of these alternatives, three suitable ANN model configurations were identified and adopted in this study for comparison. They are:

- multilayer feedforward network (MLFN),
- partial recurrent neural network (PRNN), and
- time delay neural network (TDNN).

2.3.1. MLFN

Presented in Figure 2 is a generic structure of a MLFN designed for rainfall forecasting. The output nodes of the network are the rainfall during the next time step, which contain N elements, representing the spatial locations of rainfall. For example, if forecasting rainfall at 16 points in space is needed, N is equal to 16. The number of hidden nodes, H, which defines the complexity of the network, is a key variable to be estimated. Note that the number of hidden layers can be more than one. The input layer contains k sets of input nodes. The k is referred to as the lag of the network and is another key variable to be determined.

2.3.2. PRNN

When a MLFN is used to model rainfall time series, the lag for the MLFN needs to be determined. The determination of the lag may involve a lengthy process of trial and comparison. To overcome this problem, a partial recurrent network (PRNN) is identified. The main feature of

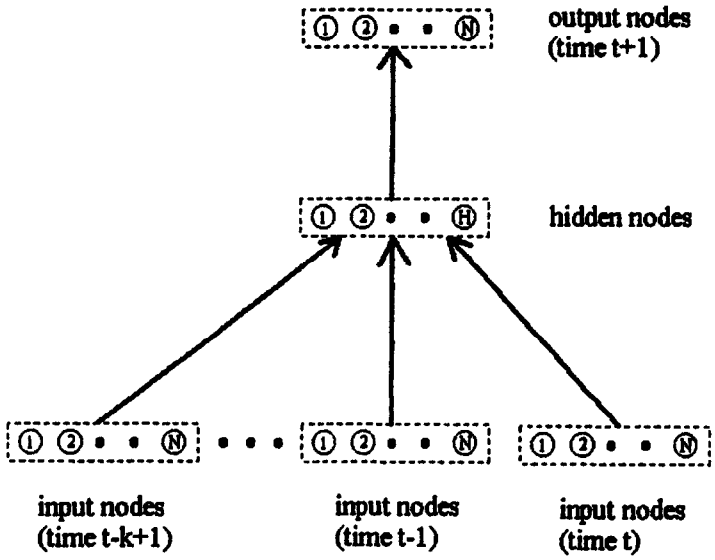

Figure 2. A MLFN for rainfall forecasting.

a PRNN is the inclusion of a set of feedback connections which allow information to pass backward. The recurrence created by the feedback connections lets the PRNN retain information from previous time steps. The temporal structure of the rainfall series is internally represented by the feedback connections. As such, the lag is not required to be specified explicitly.

The most popular PRNN, called the Elman network [4], was adopted in this study. The Elman network was originally developed for learning temporal language structures; it was subsequently found applicable for modelling other types of time series. Shown in Figure 3 is the structure of an Elman network designed for rainfall forecasting. Note that the Elman network includes a set of context units to receive feedback signals. The function of the context units is to store information from previous time steps. To achieve this aim, the context units make a copy of the activation of hidden nodes in the previous time step. Thus, at time t, the context units have some signals of the state of network at time $t-1$. As a result, the whole network at a particular time depends on an aggregate of previous states as well as on the current input. The key variable for an Elman network is the number of hidden nodes, H.

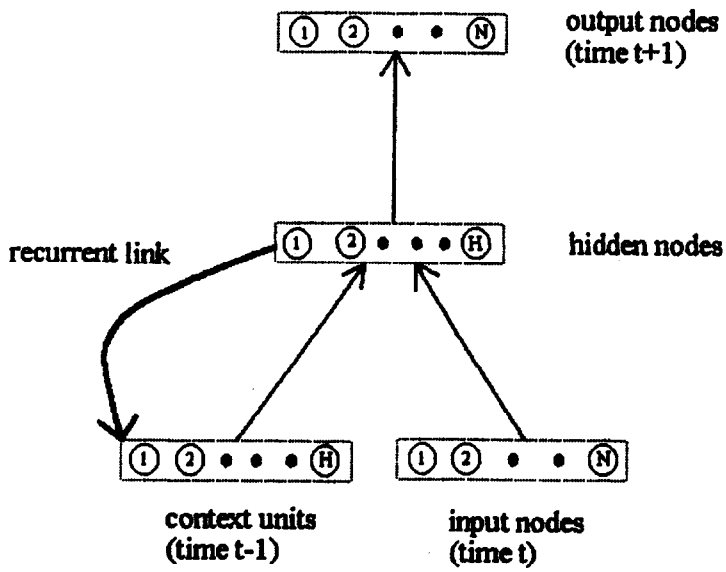

Figure 3. An Elman network for rainfall forecasting.

2.3.3. TDNN

A rainfall time series usually contains local features, such as bursts of heavy rain between periods of prolonged low intensity rainfall. These local features do not have a fixed position in time, rendering the prediction of their occurrence extremely difficult. A time delay neural network (TDNN) was adopted to handle this problem. The TDNN was originally developed by Waibel [5] for phoneme recognition. The main feature of a TDNN is the recognition of local features within a larger pattern, independent of the positions of the local features. The TDNN is essentially a feedforward network, but the connections between layers are modified. Essentially, a layer is divided into several groups and then each group separately connected to the next layer. This method reduces the number of parameters and enables local short duration features be formed at the lower layer and more complex longer duration features at the higher layer.

As an illustration, the TDNN shown in Figure 4 contains four time frames as input and three time frames in the hidden layer. Two time frames are combined to form a "window" to represent a duration in time. There are totally three windows in the input, corresponding time delays of 1, 2 and 3 (as shown in dotted lines). Each node in the hidden layer is connected to a window of two time frame of input nodes. The output is obtained by integrating (summing) the information over three time frame windows in the hidden layer.

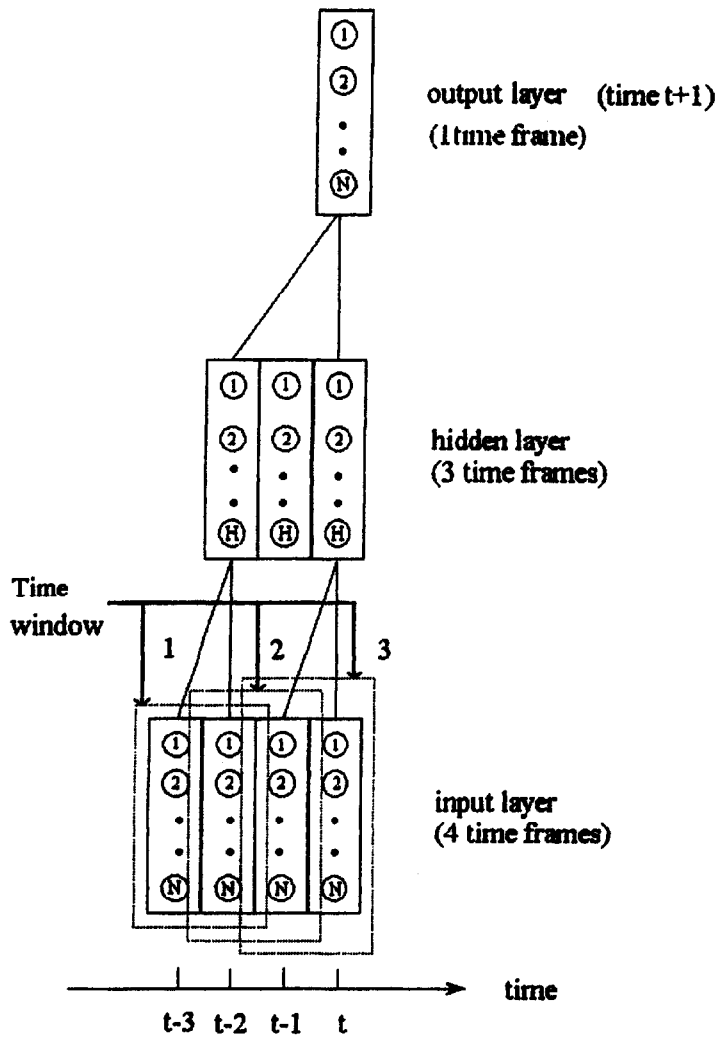

Figure 4. A TDNN with three moving windows at inputs.

With this network architecture, the hidden nodes are able to detect local features within the range of the three delays. A shift in position of the features at the input can be detected by the hidden nodes and subsequently by the output nodes.

The main variables of a TDNN need to be defined are

(1) numbers of time frames in the input and hidden layers,
(2) window size (the time delay), and
(3) the number of hidden nodes.

3. APPLICATION TO A CATCHMENT

3.1. The Study Catchment

The Upper Parramatta River Catchment, shown in Figure 5, is used as a case study for development of the alternative ANNs. The catchment is located in the western suburbs of Sydney, with a catchment size of about 112 km^2. Within the catchment, the dominant land use is typical of urban environments with a mix of residential, commercial, industrial and open space (parkland) areas.

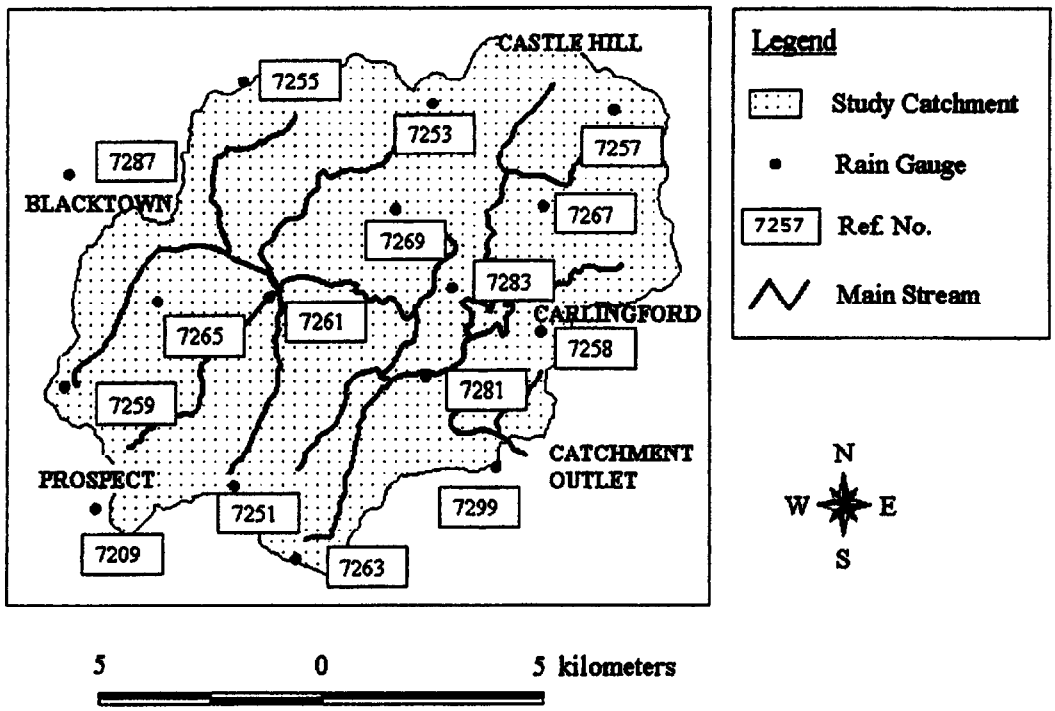

Figure 5. The Upper Parramatta River catchment.

Considerable development has occurred within the catchment over the past two decades which has lead to increases in estimates of the peak level for frequently-occurring floods. To mitigate the social and economic losses associated with floods in this catchment, the Upper Parramatta River Catchment Trust (UPRCT) was instituted in 1989 with the task of managing flood mitigation within the catchment, amongst other duties.

There are 16 (16) telemetered rain gauges within the Upper Parramatta River Catchment; locations of these gauges are shown in Figure 5. The majority of these gauges have been installed by the UPRCT since its formation. Consequently, long-term records are not available from these gauges.

Rainfall amounts during 15 minute intervals at the 16 rain gauges were obtained from January 1991 to September 1996. During this period, 34 storms occurred with daily rainfall total greater than 20 mm, with 1749 rainfall amounts at each site.

3.2. Methodology

The primary aim of developing an ANN is to generalise the features of the rainfall time series. If an ANN properly learns the features of the data, then the ANN is said to achieve good generalisation. Depending on the complexity of the network, however, an ANN may suffer from either underfitting or overfitting the training data. An ANN that is not sufficiently complex can fail to detect fully the features in a complicated data set, leading to underfitting. An ANN that is too complex may fit the noise, not just the features, leading to overfitting.

A popular technique to achieve generalisation is the early stopping method presented by Sarle [6]. According to the early stopping method, the data was split into three sets, namely a training set, a monitoring set, and a validation set. The training set was used to train the network, whereas the monitoring set was used to monitor the performance of the network at regular intervals during training. Training stopped when the error, when the model is applied to the monitoring set, reached a minimum. The validation set is used for final evaluation of the network performance.

The 34 storm events of this study were thus divided into three data sets:

- training set—16 storms with a total of 748 rainfall periods (each of 15 minutes),
- monitoring set—eight storms with a total of 376 rainfall periods, and
- validation set—ten storms with a total of 625 rainfall periods.

The maximum epoch for training was set at 1000. An epoch was defined as a complete sweep through the training patterns; the weights of a network were updated after each epoch. Therefore, a maximum epoch of 1000 means that the weights were allowed to update at most 1000 times. During training, the networks were checked at every 100 epochs. Training was stopped when the error in the monitoring data reached its lowest value, or the training reached the maximum epoch, whichever came first. Finally, the networks were evaluated against the validation data.

Where appropriate, a sigmoid activation function was adopted for the hidden nodes, whereas a linear activation function was used for the output nodes. The use of a sigmoid function was to enable nonlinearity of the network. The sigmoid function, however, was not adopted for the output nodes because it would force the output to be bounded between 0.0 and 1.0; this would require scaling of the output variable by a known maximum value. This was not appropriate for rainfall forecasting because it was undesirable to set *a priori* a maximum rainfall value for the data. To overcome this situation, an identity (linear) function was used instead.

To achieve better performance and faster convergence in training, the data were transformed with the log function

$$y = 0.5 * \log_{10}(x + 1), \tag{5}$$

where

x are the rainfall data, and
y are the data after transformation.

The normalised mean squared error (NMSE), defined as follows, was chosen as the performance indicator:

$$\text{NMSE} = \frac{\sum_o \sum_p (d_{op} - y_{op})^2}{\sum_o \sum_p (d_{op} - \bar{d}_{op})^2} \approx \frac{1}{OP\sigma^2} \sum_o \sum_p (d_{op} - y_{op})^2, \tag{6}$$

where

O is the total number of output nodes,
P is the total number of patterns,
d_{op} are the transformed target outputs,
y_{op} are the transformed network outputs, and
σ^2 is the variance of the transformed target outputs.

In essence, the NMSE is the sum of squared errors (SSE) normalised by the number of testing patterns over all output nodes and the estimated variance of the data. The use of NMSE enables a comparison of results with different length of testing patterns. Note also that a value of NMSE = 1 corresponds to simply predicting the average.

4. TEST RESULTS AND DISCUSSIONS

During the development of the alternative ANNs, various network configurations were attempted in order to determine the effect of two key variables, which are:

- the lag of network, and
- number of hidden nodes.

For the MLFN, networks with lags 1, 2, 3, and 4 were attempted. In addition, the numbers of hidden nodes tried were 2, 4, 8, 16, 24, 32, 64, and 128. Networks with two layers of hidden nodes were also attempted. For the Elman network, the order of lag was fixed at 1 because the network would learn the temporal structure implicitly. The numbers of context units tried were 2, 4, 8, 16, 24, 32, and 64. Finally, for the TDNN, networks with 2, 3, and 4 input windows were attempted.

Shown in Table 1 are details of the eight best networks as determined by analysis of the results from testing all the above network configurations; the selection was based on the minimum NMSE for the validation data set. Shown in each row of Table 1 is the result of the best network of a given lag. For example, the best lag-1 MLFN was the one which consisted of 24 hidden nodes. Similarly, the best lag-2 MLFN was the one which consisted of eight hidden nodes, and so on.

In general, all three types of networks showed comparable performance. The NMSE of the validation samples for all networks were in the range of 0.63 to 0.67, with only small differences

Table 1. Comparison of alternative networks. Note: The network configuration is denoted by three figures (x-y-z), where x = no. of input nodes, y = no. of hidden nodes and z = no. of output nodes.

Network	Training (NMSE)	Monitoring (NMSE)	Validation (NMSE)	Stopping Epoch	Training Error at 1000 Epoch (NMSE)
MLFN Lag 1 (16-24-16)	0.50	0.68	0.64	200	0.49
MLFN Lag 2 (32-8-16)	0.51	0.69	0.66	100	0.47
MLFN Lag 3 (48-4-16)	0.48	0.69	0.67	700	0.47
MLFN Lag 4 (64-2-16)	0.52	0.71	0.65	200	0.49
Elman (16-4-16)	0.49	0.67	0.64	300	0.48
TDNN Lag 2 (32-16-16)	0.50	0.67	0.63	100	0.41
TDNN Lag 3 (32-16-16)	0.50	0.69	0.64	100	0.41
TDNN Lag 4 (64-32-16)	0.51	0.69	0.65	100	0.40

between the networks. An explanation of the comparable performance was that the networks shown in Table 1 were developed to their optimal complexities. For example, the lag-1 MLFN required as many as 24 hidden nodes to achieve an optimal complexity, whereas the more complicated lag-4 MLFN only required two hidden nodes to offset the large number of parameters introduced by the higher order of lag. The reduction in the number of hidden nodes with an increase in lag, and vice versa, might indicate that the existence of an optimal complexity of network for the problem being considered, given the data available.

Another interesting aspect of the results was that the networks with a lower lag had a slightly better performance than those of a higher lag. This might suggest that the rainfall series did not have long-term dependence structures. The short-term memory characteristics of the rainfall time series helped to explain why the simple MLFN had comparable performance with the more sophisticated TDNN and Elman networks.

Shown in Figures 6 and 7 are comparisons of the three types of networks considered for forecasting of rainfall depth one time-step ahead at a gauge during two validation storm events. It was noted from Figure 6 that all three types of networks produced reasonable forecasts of the rainfall one time step ahead. For this storm, the networks made a good forecast of the decrease in rainfall rate in the storm evolution. However, none of the networks forecast the peak rainfall rate well. This might be attributed to the fact that the peak occurred randomly, and was preceded by low rainfall rates.

Similar observations were noted from Figure 7. For this storm, however, there was a steady rate of increase as well as decrease of rainfall rate in the storm evolution, the networks were able to make good predictions.

For this pattern recognition approach, an improvement in prediction is expected through the collection of more data for training the model. Additionally, in order to predict the occurrence of the peak rainfall rate more accurately, it may be necessary to identify more control variables, such as wind speed and direction, and incorporate them into the network input.

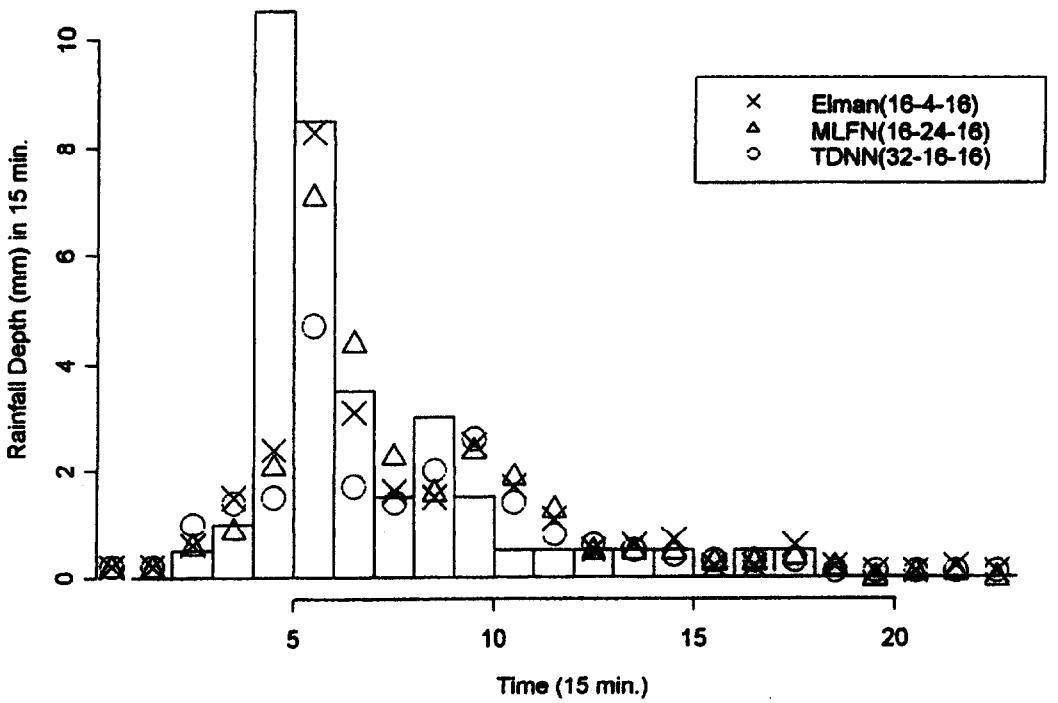

Figure 6. Forecasting rainfall at gauge no. 7253 for the storm event on January 2, 1996.

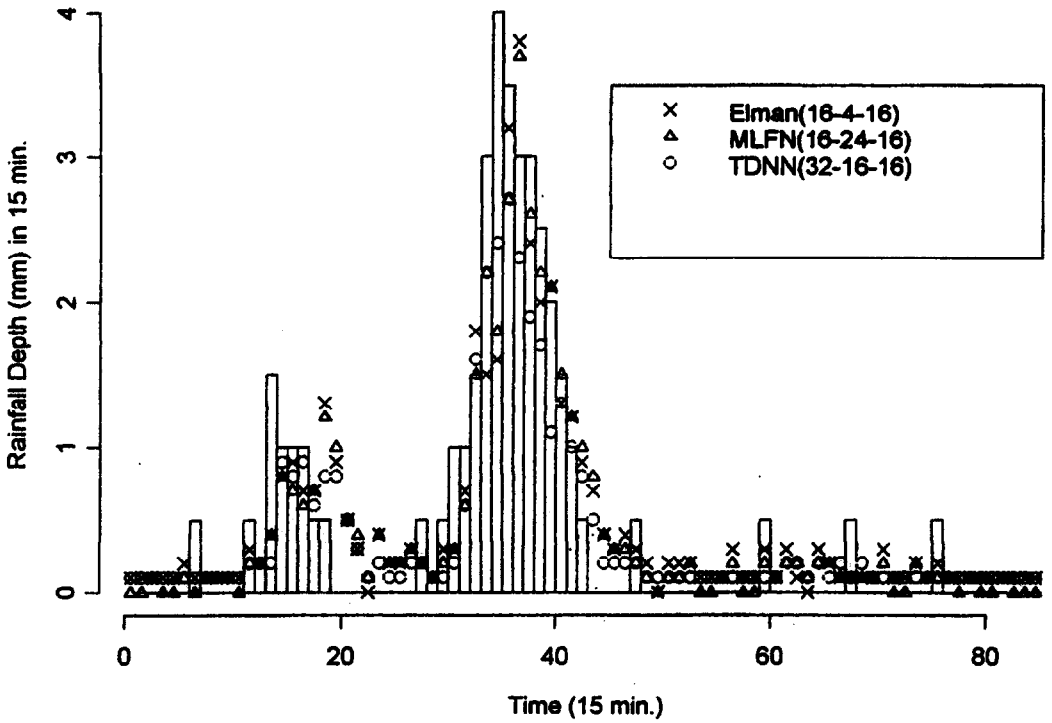

Figure 7. Forecasting rainfall at gauge no. 7253 for the storm event on January 6, 1996.

5. CONCLUSION

Rainfall forecasting using ANNs has been the focus herein. Three types of ANNs suitable for this task were identified, developed, and compared; these networks were

- multilayer feedforward neural network (MLFN),
- Elman partial recurrent neural network (Elman), and
- time delay neural network (TDNN).

All the above alternative networks could make reasonable forecast of rainfall one time step (15 minutes) ahead for 16 gauges concurrently.

In addition, the following points were observed.

- For each type of network, there existed an optimal complexity, which was a function of the number of hidden nodes and the lag of the network.
- All three networks had comparable performance when they were developed and trained to reach their optimal complexities.
- Networks with lower lag tended to outperform the ones with higher lag. This indicates that the 15-min. rainfall time series have very short term memory characteristics.

REFERENCES

1. K. Hornik, M. Stinchcombe and H. White, Multilayer feedforward networks are universal approximators, *Neural Networks* **2**, 359–366, (1989).
2. M. French, W. Krajewski and R.R. Cuykendall, Rainfall forecasting in space and time using a neural network, *Journal of Hydrology* **137**, 1–31, (1992).
3. K.L. Hsu, V. Gupta and S. Soroshian, Artificial neural network modeling of the rainfall-runoff process, *Water Resources Research* **31** (10), 2517–2530, (1995).
4. J.L. Elman, Finding structure in time, *Cognitive Science* **14**, 179–211, (1990).
5. A. Waibel, Modular construction of time-delay neural networks for speech recognition, *Neural Computation* **1**, 39–46, (1989).
6. W.S. Sarle, Stopped training and other remedies for overfitting, *Proceedings of the 27th Symposium on the Interface of Computing Science and Statistics*, 352–360, (1995).