# Neural Networks for River Flow Prediction

Nachimuthu Karunanithi[1], William J. Grenney[2], Member, ASCE,
Darrell Whitley[3] and Ken Bovee[4].

**Abstract**

The surface water hydrographs of rivers exhibit large variations due to many natural phenomena. One of the most commonly used approachs for interpolating and extending streamflow records is to fit observed data with an analytic power model. However such analytic models may not adequately represent the flow process because they are based on many simplifying assumptions about the natural phenomena that influence the river flow. This paper demonstrates how a neural network can be used as an adaptive model synthesizer as well as a predictor. Issues such as selecting an appropriate neural network architecture and a correct training algorithm as well as presenting data to neural networks are addressed using a constructive algorithm called Cascade-Correlation algorithm. The neural network approach is applied to the flow prediction of the Huron River at the Dexter sampling station near Ann Arbor, Michigan. Empirical comparisons are performed between the predictive capability of the neural network models and the most commonly used analytic nonlinear power model in terms of accuracy and convenience of use. Our preliminary results are quite encouraging. An analysis performed on the structure of the networks developed by the Cascade-Correlation algorithm shows that the neural networks are capable of adapting their complexity to match changes in the flow history and that the models developed by the neural network approach are more complex than the power model.

[1]Post Doc., Bell Communications Research, 445 South Street, Morristown, NJ 07692. Tel: (201) 829-4466, Fax: (201) 829-5888, Email: *karun@faline.bellcore.com.*

[2]Prof., Civil Eng., Utah State Univ., Logan, UT 84322.

[3]Assoc. Prof., Dept. of Comp. Sci., Colorado State Univ., Fort Collins, CO 80523.

[4]Hydrologist, National Ecology Research Center, Fort Collins, CO 80525.

# 1 Introduction

The motivation for this study was to estimate flows at an ungauged site on the Huron River, Michigan, to provide data for assessing how future alterations to flows would affect habitat for smallmouth bass (Bovee 1992). In order to quantify habitat variability, it was necessary to relate previously recorded habitat samples taken over a period of several years to the streamflow occurring at the time of each sample. U.S. Geological Survey stream gauging stations located about 30 $km$ upstream and 20 $km$ downstream from the sampling site have continuous records since 1960. Two other gauging stations in the vicinity were discontinued in 1977 and 1982.

Flows in streams and rivers are complex processes that are influenced by many factors such as the watershed topography, vegetation cover, soil types, channel characteristics, groundwater aquifers, precipitation distribution, snow melt, rural and urban activities, etc. Engineering project design and environmental impact analysis often require the estimation of streamflows, or their statistical properties, at ungauged sites. A variety of methods have been developed for this purpose including parametric models for synthesizing streamflows from basin characteristics and meteorological data (Beven 1989, Chow 1964), and empirical models for interpolating or extrapolating data from gauged sites (Chow 1964; Crawford and Linsley 1966). Parametric models are expensive and time consuming to apply, and are normally employed only when insufficient data exist to utilize a simpler empirical method.

Sufficient data were available in this situation to apply the most commonly used empirical method for interpolating and extending streamflow records: a power model fit to the log transform of the flow data (Chow 1964). This analysis was performed for the Huron River with a spreadsheet using 13 years of daily data. A short period of record from a discontinued gauge was used to test the power model. Bovee (1992) attempted

to improve the results by refining the analysis in two ways. First he smoothed the data by taking averages over every five-day period ("five-day non-overlapping averages") to account for the travel time between stations. Under this smoothing, for example, days 1 through 5 are used to produce 1st average value, and days 6 through 10 for the 2nd average value and so on. Second he divided the data into twelve monthly data sets and fit a separate power curve to each data set using the spreadsheet program. The data manipulation involved a great deal of manual effort and provided many opportunities for mistakes. The objective of this preliminary research was to examine an alternative empirical method with the hope of achieving three things: 1) greater accuracy, 2) increased convenience for the user to develop an appropriate model for the flow history and 3) evaluate various data representations.

Recently, neural networks have been successfully applied to many applications in civil engineering (Cheu *et al.* 1991; Moselhi *et al.* 1991; Ramirez and Arghya 1991) and structural engineering (Kamarthi *et al.* 1992; Furuta *et al.* 1991; Hajela *et al.* 1991; Xihui *et al.* 1991) as well as in many other fields (Weigend *et al.* 1990; Karunanithi *et al.* 1992a & 1992b; Karunanithi 1992c). Feed-forward neural networks are most widely used in these applications. The networks are trained using the standard error back-propagation algorithm. However, one major limitation of this training algorithm is that the architecture of the network has to be fixed in advance. This means that the end user must design a suitable architecture by a costly trial-and-error approach. If the architecture is too small the network may not have sufficient degrees of freedom to correctly learn the flow process. On the other hand, if the network is too large then it may not converge during training or it may overfit the data and memorize the flow history rather than generalize it. This paper evaluates the applicability of the neural network approach using the Cascade-Correlation algorithm developed by Fahlman and Lebiere (1990). The Cascade-Correlation algorithm is a constructive algorithm that can automatically synthesize a suitable network architecture as part of its training process. We selected the

Cascade-Correlation algorithm for this study because our objective was to evaluate the predictive capability of different neural network models rather than concentrate on issues such as selection of a suitable architecture for the network and convergence of the learning algorithm.

Our preliminary results suggest that the neural network approach is capable of providing a more accurate prediction compared to the power model. Our results also suggest that the neural network approach may not require any transformation or smoothing on the data set. An analysis performed on the structure of the networks developed by the Cascade-Correlation algorithm shows that the neural networks are capable of adapting their complexity to match changes in the flow history.

# 2 Overview of Neural Networks

Artificial Neural Networks are a computational metaphor which was inspired by studies of the brain and nervous systems in biological organisms. They represent highly idealized mathematical models of our present understanding of such complex systems. Typically, a neural network consists of a set of layered processing units and weighted interconnections. Neural networks operate on the principle of learning from a training set. There exists a variety of neural network models and learning procedures. (Readers not familiar with neural networks may refer to Lippmann (1987); or any introductory book on neural networks such as Rumelhart *et al.* (1986) for more details.) Two well-known classes of neural networks that can be used for prediction applications are: *feed-forward networks* and *recurrent networks*. In a feed-forward network the weighted connections feed activations only in the forward direction from the input layer to the output layer. On the other hand, in a recurrent network additional weighted connections are used to feed previous activations back into the network. In this paper we focus on only feed-forward networks. Learning in neural networks involves adjusting the weights of interconnections. The most

commonly used training algorithm for feed-forward networks is the *back-propagation* algorithm by Rumelhart, Hinton and Williams (1986). (A clear overview of the feed-forward network and the back-propagation algorithm can be found in Kamarthi *et al.* (1992) in an earlier issue of this journal.) The back-propagation algorithm is a gradient descent method in which weights of the connections are updated using partial derivatives of error with respect to weights. However the standard back-propagation algorithm can train only on a network of predetermined size. Both the accuracy of predictions and a network's learning ability can be severely affected if the architecture is not suitable. This implies that for a given application the problem of specifying a suitable architecture must be addressed first. Finding a suitable network architecture can be a very time consuming exercise. Instead, we use the Cascade-Correlation architecture algorithm (Fahlman and Lebiere 1990) which can synthesize an appropriate architecture and train the neural networks simultaneously. This algorithm is summarized next.

## 2.1 The Cascade-Correlation Algorithm

The Cascade-Correlation algorithm is an efficient constructive training algorithm developed by Fahlman and Lebiere (1990). This algorithm combines the idea of incremental architecture and learning in its training procedure. In brief, training starts with a minimal network consisting of an input and an output layer. Note that this minimal network (without a hidden unit) is a one-layered network and that it is equivalent to a linear model. If the training algorithm can no longer reduce the residual error, then it stops this phase of training, and enters the next phase for training a potential hidden unit. The potential hidden unit has associated connections (or, weights) from the input layer and all preexisting hidden units, but not towards the output layer. Weights associated with the potential hidden units are optimized by a gradient ascent method so as to maximize the correlation between its output and the residual error of the network. When

a potential hidden unit is trained, weights associated with the output layer are kept unchanged. Once a potential hidden unit is added to the network, it becomes a new hidden unit and its incoming weights are frozen for the rest of the training period. After installing a hidden unit, the training updates weights of all connections that directly feed the output layer. This dynamic expansion of the network continues until the problem is successfully learned. Thus the Cascade-Correlation algorithm automatically constructs a suitable network architecture for a given problem. Other major advantages in using the Cascade-Correlation algorithm include i) more consistency in solving problems and ii) an order of magnitude faster learning than the standard back-propagation algorithm (Whitley and Karunanithi 1991).

A typical feed-forward network trained by the back-propagation algorithm (BP-network) and a network synthesized by the Cascade-Correlation algorithm (CASCADE-network) are shown in Figures 1(a) and 1(b). Note that the CASCADE-network in Figure 1(b) has direct connections between the input layer and the output layer as well as lateral connections to each new hidden unit from all previously added hidden units. In the CASCADE-network each hidden unit is equivalent to a separate hidden layer.

. . . . . .

**Insert Fig. 1 about here.**

. . . . . .

## 2.2 The Proposed Network Structure

Most neural networks are constructed using the widely known *sigmoidal unit* with a logistic activation function. The output response of a typical sigmoidal unit is bounded over a 0.0 to 1.0 range. This means that the user has to scale the output variable of

the problem using a known maximum value. Especially in applications such as the river flow prediction it may not be possible to set a priori a reasonable maximum value. Use of an inaccurate maximum value for scaling the output variable can severely affect the predictive capability of the neural network models. A solution to the scaling problem is presented next.

The input-output responses of processing units are shown in Figure 2. Figure 2(a) shows how a typical processing unit works. First, *net* is computed as a sum of weighted inputs. Then *net* is transformed to an output value by applying an *activation function*. Figure 2(b) shows the activation function of a sigmoidal unit, which is the most widely used form in neural network models. The output of a sigmoidal unit is given by:

$$\text{Output} = \frac{1}{1 + e^{-net}} \tag{1}$$

where *net* is the weighted input from all incoming units.

.....

**Insert Figure 2 about here.**

.....

Figure 2(c) represents the solution in terms of a clipped linear function. Instead of using a sigmoidal unit in the output layer one can use a *linear unit* with a clipped linear function. The output of a clipped linear unit is given by:

$$\text{Output} = \begin{cases} \text{net} & \text{if } net > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

By replacing the sigmoidal output unit with a clipped linear unit the network can produce any positive value as its output. However, one difficulty with this clipped linear function is that it does not have a continuous derivative for $net \leq 0$. The derivative must be continuous in order for these learning algorithms to work correctly. A simple alternative

for the above clipped linear function is to make it a $1/net$ type function for $net < 0$. This will force the output of the linear unit to drop to zero very quickly when $net < 0$. This type of function can provide a well behaved, non-zero derivative for all parts of the activation function while adding a negligible value to the output for $net \leq 0$. The actual activation function examined in this paper is given by,

$$\text{Output} = \begin{cases} \text{net} & \text{if } net > 0 \\ 1/(a - b \cdot \text{net}) & \text{otherwise} \end{cases} \tag{3}$$

where $a$ and $b$ are constants. The derivative of this modified function is given by,

$$\frac{\partial \text{Output}}{\partial net} = \begin{cases} 1 & \text{if } net > 0 \\ b/(a - b \cdot \text{net})^2 & \text{otherwise} \end{cases} \tag{4}$$

By setting $b = a^2$ the derivative reduces to 1.0 at $net = 0$. However it is necessary to select a reasonable value for $a$ so that the "output" (eqn. 3) rapidly drops to zero when $net < 0$. In our experiments we used the following values: $a = 100.0$ and $b = 10000.0$.

In the following application, the flow rate of the Huron river varied from a minimum of 20 $cfs$ up to a maximum of 3500 $cfs$. We mapped the hydrograph to a small "open range" by dividing each value by 100. This smaller range improved the learning speed of the neural network. This *1-to-1* mapping affects neither the accuracy of prediction nor the precision of the output.

## 2.3 Training Neural Networks

Neural networks are trained with a set of typical input/output pairs called the *training set*. The final weight vector of a successfully trained neural network represents its knowledge about the problem. In general, it is assumed that the network does not have any a priori knowledge about the problem before it is trained. So at the beginning of training the network weights are initialized with a set of random values. In our experiments the

network weights were initialized with a set of uniform random values drawn between -1.0 and 1.0. During training the weights are adjusted so as to reduce the residual error of the training set. Since the Cascade-Correlation algorithm adjusts not only the network's weights but also the structure of the network (i.e., the number of weights in the network) care must be exercised to see that the final network is neither too complex nor too simple. In this algorithm the complexity of the network is inversely related to the value of a unit free parameter called "Error_Index_Threshold". This parameter represents the *root mean square* of sum squared residual error of the training set normalized by the standard deviation of the training outputs. Thus by controlling this parameter we can constrain the number of hidden units added to the network. After examining a few values, we settled for a value of 0.15 for the Error_Index_Threshold because the resulting networks provided a better prediction results than other values.[5]

# 3 Flow Estimation for The Huron River

## 3.1 Problem Definition

Bovee (1992) conducted a study to assess the impact of altering flows in the Huron River, Michigan, on the habitat for smallmouth bass. In order to quantify habitat variability, it is necessary to relate previously recorded habitat samples to the streamflow occurring at the time each sample was taken. Figure 3 is a schematic diagram of the principal features in the subbasin. The segment of river used for the habitat study extends from Zeeb Road, about 20 $km$ upstream from Ann Arbor, to Bell Road about 15 $km$ above Zeeb Road.

[5]However, a default value of 0.20 seems to be fairly robust across many other problems that we have studied.

. . . . . .

**Insert Figure 3 about here.**

. . . . . .

The town of Dexter marks the mid-point of the study area and delineates the "upper" section from the "lower" basin. Separate fish population and habitat data have been collected in both the upper and lower sections to provide spatial as well as temporal relationships. Mill Creek is the only significant tributary in the subbasin, and it enters the Huron in the lower section. Four U. S. Geological Survey gauging stations are located in the area as shown by the boxes in Figure 3.

The periods of record since 1960 for each of the gauging stations are indicated in Figure 4. The solid lines in the figure represent the periods of record. The dashed lines in Figure 4 represent the missing data.

. . . . . .

**Insert Figure 4 about here.**

. . . . . .

The year 1960 was selected as the starting point for the analysis in order to provide continuous concurrent data streams at four sites. The gauges at Hamburg ($H$) and Ann Arbor ($A$) have continuous records from 1960 to the present. The gauge at Dexter ($D$) has a continuous record from 1960 to 1972, and again from 1976 to 1977. The gauge at Mill Creek ($M$) has a continuous record from 1960 to 1982.

The problem was to estimate the flows in the upper and lower sections on specified dates between 1978 and 1990 given the flows at Hamburg and Ann Arbor on those dates. Analyses of the data during the period of concurrent record indicated that the flow in the lower section below the confluence with Mill Creek could be assumed to be equal to the flow at Ann Arbor. However, the flow above Mill Creek could not be assumed to

be equal to the flow at Hamburg because of significant ungauged lateral inflow between Hamburg and Dexter. A method was needed to estimate the flow at Dexter based on the observed flows at Hamburg and Ann Arbor.

## 3.2 Application of The Power Model

The following two-station power model was applied (Chow 1964).

$$D = \beta_0 A^{\beta_1} H^{\beta_2} \tag{5}$$

where,

$D$ = the discharge measured at Dexter,

$A$ = the discharge measured at Ann Arbor,

$H$ = the discharge measured at Hamburg, and

$\beta_0, \beta_1$ and $\beta_2$ are model coefficients.

The corresponding logarithmically transformed regression model is given by:

$$\log(D) = \log(\beta_0) + \beta_1 \log(A) + \beta_2 \log(H) \tag{6}$$

In experiment 1, coefficients were estimated by least square regression of Equation 6 to daily flows over the period of concurrent record, 1960-1972, using the QuattroPro spreadsheet program (Borland 1991). (However, it should be noted that this training period may not be sufficiently long enough to capture low frequency variations in the flow process.) The model was tested by comparing predicted values at Dexter, $\hat{D}$, with the observed values $D$ during the two year period 1976 and 1977. Figure 5 is a graph of the observed flows at Dexter (indicated by the solid line) and the predicted values (indicated by the dashed line). The time scale on the horizontal axis is in terms of days since the beginning of water year 1976. Water years run from October 1 to September 30, and are identified by the calendar year in which the water year ends. For example in

the Figure 5, day 1 is October 1, 1975, day 32 is November 1, 1976, day 95 is January 1, 1976, and so on.

. . . . . .

**Insert Figure 5 about here.**

. . . . . .

The model provides an adequate estimate of flows during the test period except for the extreme events around day 175 which are considerably underestimated.

Residual errors were quantified in two ways:

1) The square error ($se$)

$$se_i = (\hat{D}_i - D_i)^2 \tag{7}$$

2) The relative error ($re$)

$$re_i = \left| \frac{(\hat{D}_i - D_i)}{D_i} \right| x100 \tag{8}$$

where $i$ is an index on the observation in the data set. The mean square error ($mse$) is defined by:

$$mse = \frac{1}{n} \sum_{i=1}^{n} se_i \tag{9}$$

Both the relative error ($re$) and square error ($se$) measures presented in this discussion are based on the flow values and not on the log transforms. The mean square error of the log transform is minimized, of course, for the regression of Equation 6, and may produce biased estimators (McCuen, Leahy and Johnson 1990). However, we are interested in the comparison of the predicted flow with the observed flow, and not directly in the accuracy with which the polynomial, Equation 6, fits the transformed data. Likewise the mean relative error ($mre$) is defined by:

$$mre = \frac{1}{n} \sum_{i=1}^{n} re_i \tag{10}$$

The square error ($se$) and the relative error ($re$) provide different types of information about the predictive capabilities of the model. Figure 6 is a plot of the $se$ for each observation and demonstrates that this measure is more sensitive to errors at high flow than at low flow. The $se$ is a good measure for indicating goodness-of-fit at the high flows. Figure 7 is a plot of the $re$ for observation and provides a more balanced perspective of the goodness-of-fit at moderate flows. The $mse$ and the $mre$ are 11,763 and 11.9 respectively for the two year test period.

. . . . . .

**Insert Figures 6 and 7 about here.**

. . . . . .

Figure 6 shows that the largest prediction errors are associated with the high runoff events around Day 150. This error may be related to snowmelt in the headwaters of the Huron. The only meteorological station in proximity to the Dexter gauge is at the University of Michigan campus at Ann Arbor. Rainfall is not distinguished from snowfall at this station, so adding total precipitation to a power model would not significantly improve its accuracy. So an analysis was made to see if the addition of Mill Creek to the power model would improve the predictions for the test period. This was also just a matter of curiosity because the Mill Creek records ends in 1982 and could not be used to estimate flows at Dexter during the required period from 1983 through 1990. As it turned out, adding Mill Creek to the power model increased the $mse$ and the $mre$ by about 7% for the two year test period.

Bovee (1992) attempted to improve the results by refining the analysis in two ways. First, in experiment 2, he smoothed the data by using five-day averages to account for the travel time between gauging stations (i.e., the flow rate was averaged over every

consecutive five day period to produce a series of five-day average values). The power model was fit to the 13 years of smoothed data and then tested against smoothed data for the two year test period. The $mse$ was reduced to 6,372 and the $mre$ was reduced to 7.5.

Second, in experiment 3, he divided the smoothed data into 12 monthly data sets. That is, he put all of the January data from the 13 years into one data set, all of the February data into another set, and so on. Then the power model was separately fit for each data set using the spreadsheet. The mothly models were tested by using appropriate monthly data sets corresponding to the two year test period. The resulting $mse$ and $mre$ were 12,598 and 8.5 respectively. It is interesting to note that the error measurements actually increased when the number of models was increased from one to twelve. An examination of the data for the month of March provided an explanation. The 13 years of March flows used for the regression were relatively low compared to the unusually (record) high runoff in March 1976 of the test data. Thus the power model developed solely from previous March data was a worse predictor of March 1976 flows than the power model developed from the aggregation of flows from all months.

## 3.3 Application of Neural Network Models

The neural network approach was evaluated using three testing experiments (applied to the power model) in order to compare the two techniques in terms of accuracy and convenience of use. Two neural network architectures were evaluated in this study. The networks are shown in Figure 8. The output layers of these networks were constructed using a clipped linear unit (proposed in the previous section) while the hidden layers were constructed using the sigmoidal unit. Figure 8(a) shows a standard feed-forward network with three inputs and one output (NN1). The inputs $H$, $M$ and $A$ represent the discharges recorded at Hamburg, Mill Creek and Ann Arbor respectively. The output $D$

represents the corresponding discharges observed at the Dexter gauge. As in the power model, the values of $H, M, A$ and $D$ represent daily values for the experiment 1 and five-day averages for the experiments 2 and 3.

. . . . . .

**Insert Fig. 8 about here.**

. . . . . .

The five-day non-overlapping averaging used for the experiments 2 and 3 acts as a linear filter for reducing abrupt fluctuations in the hydrograph. However, this type of smoothing may or may not be appropriate. Alternatively, we can avoid this type of smoothing and let the network do its own preprocessing. So we modified the structure of the NN1 network to feed daily values on the input side instead of five-day averages. The resulting network, NN2, is shown in Figure 8(b). The five-day window consists of $(t-4, t-3, t-2, t-1, t)$ where $t$ represents the current day, $t-1$ the previous day, $t-2$ the day before the previous day and so on. However, for experiments (2) and (3), the outputs for the network NN2 had the same values as that of NN1. Since we wanted to train the NN2 model with the same number of training patterns as that of NN1, we presented the data in a different format: On the input side we used a five-day sliding window such that the 1st instance of the window is formed by days $t$ through $t-5$, and the 2nd instance by days $t+1$ through $t-4$ and so on. On the output side we used the corresponding five-day moving averages rather than the five-day non-overlapping averages.

Since the Cascade-Correlation algorithm is stochastic in nature, it is possible for the algorithm to produce networks with varying size and different final weight vectors[6]. Table 2 illustrates variations in the size of the networks. Such a variation in the size

[6]The variations are due to the random number generator used to initialize the weight vectors of the network and the incoming connection weights of candidate hidden units.

of the network may introduce variation (or uncertainty) in the prediction result. One possible way to reduce such a variation in predictions is to conduct a large number of Monte-Carlo trials and take their average (or the best value). So, for each experiment we trained both NN1 and NN2 50 times with different sets of initial weights and averaged their predictions.

The training set for the first experiment consisted of the daily flows during the concurrent period 1960 through 1972, a total of 4748 observations. The test set consisted of the daily flows during the two-year test period 1976 and 1977, a total of 731 observations. Several preliminary runs were made with and without the Mill Creek data to assess the influence of the Mill Creek data on the predictive accuracy of the neural network model. Our preliminary results show that, unlike in the power model, the neural network's predictive accuracy is not affected by the inclusion of the Mill Creek data. The $mse$ and the $mre$ are 7,272 and 7.6 respectively for the network that included the Mill Creek data, whereas, the corresponding values for the network without the Mill Creek data are 10,247 and 8.0 respectively. A similar difference in performance was observed in other experiments also. Thus including the Mill Creek data improved the predictive accuracy of the neural networks. One possible reason for this improvement in predictive accuracy may be due to the fact that the neural network is able to utilize the Mill Creek data as additional information rather than as a source of noise. Based on this observation we included the Mill Creek data in the remaining two experiments also.

Figure 9 is a graph of the observed flows at Dexter (indicated by a solid line) and the predicted values of NN1 (indicated by a dashed line) for the experiment (1). We included this graph to illustrate how a typical neural network predicts the daily flow. The predicted values of NN1 in Figure 9 represent the predictions of a network that had 16 hidden units. For comparison purpose we also included bar graphs for the $se$ and the $re$ in Figures 10 and 11 respectively.

. . . . . .

**Insert Figures 9, 10 and 11 about here.**

. . . . . .

From the graphs in Figures 5 and 9 it is not clear which model is better. However, a close examination of the predicted values during the high runoff period (from 130-th day to 180-th day) suggests that the NN1 model is able to predict the high runoffs more accurately than the power model.

The $mre$ of the NN1 model and the power model are 12.4 and 11.9, and the corresponding $mse$ are 11,365 and 11,763 respectively. A comparison of bar graphs in Figures 6 and 10 clearly suggests that the neural network model NN1 is a more accurate predictor of high runoffs than the power model. These graphs also suggest that both models have a similar predictive accuracy during low flows. We found a similar observation in other experiments as well as for the neural network model NN2.

Next, we conducted the remaining two experiments with these two network models. The training set for the second experiment consisted of the smoothed data, non-overlapping five-day averages, a total 949 points in the training set and 146 points in the test set. The training set for the third experiment consisted of the smoothed data divided into 12 monthly sets with a separate network fit to each set, as was the approach with the power model. A summary of results for all three experiments and a comparison of the predictive accuracy of these models are presented next.

## 3.4 Results

Table 1 presents a summary of prediction results. The first three rows in Table 1 (under "Testing Error") contain the errors from the testing set, whereas the remaining three rows (under "Training Error") contain the residual from the training set. The training

set error was measured by feeding the training set (or, the data set used for estimating regression coefficients) itself as the test input. The training set errors are included to indicate the accuracy of models' fit with the training data.

.....

**Insert Table 1 about here.**

.....

First, we compare the predictive accuracy of NN1 and the power model in terms of $mse$. In experiments 1 and 3 the neural network model seems to have a slight edge over the logarithmic regression model. On the other hand, in experiment 2 the logarithmic model seems to be slightly more accurate than the neural network model NN1. Since the difference in error in all three experiments are not very significant we can conclude that there is no clear winner among these competing models. However, the $mre$ measure seems to indicate that the power model is slightly more accurate than the NN1 model. This could be misleading to some extent because a model may be more accurate overall but it may not be a good predictor of high runoffs. If the hydrograph is dominated by relatively a large number of low flow measures and a few high runoffs then the model may tend to learn low flow rates more accurately than high flow rates. Thus $mse$ is considered as the appropriate measure for comparison.

Next, we compare the predictive accuracy of the NN2 model with the NN1 model and the power model. The $mre$ values of the NN2 model are higher than the two other models in most of the cases. However, the $mse$ values of the NN2 model are significantly lower than the other two models in all three experiments. This suggests two things: 1) the NN2 model is a more accurate predictor than other two models and 2) allowing the network to have a five-day window is more useful than feeding five-day averages. Another observation to be made from these results is that the neural network models are

not affected by the Mill Creek hydrograph. This suggests that, unlike the power model, the neural network models are less vulnerable to additional noise in the data.

# 4 Discussion

## 4.1 Complexity of the Models

The testing set results in the previous section suggest that the neural network models are more accurate predictors than the power model. However, it is not clear how and why neural networks predict better than the power model. In this section we provide a simple analysis on the network structure so as to provide further insights into their modeling capability. As pointed out earlier, the Cascade-Correlation algorithm develops networks of varying complexity to match the complexity of the training set. This is illustrated in Table 2 in terms of the number of hidden units used. The "Min" and "Max" values represent the minimum and the maximum number of hidden units used during 50 trials. The Standard Deviation (SD) gives an indication about the amount of variability in the number of hidden units used.

. . . . . .

**Insert Table 2 about here.**

. . . . . .

In experiment 1, the NN1 model used on the average 15.92 hidden units whereas in experiment 2 the average has decreased to 1.16. This clearly suggests that the smoothing has considerable effect on the size of the resulting nets. In experiment 3, the above values were calculated after combining the averages obtained for 12 calendar months separately. The different models associated with different individual months displayed very different complexity. The network learned the flow history of January, May, July and August

without any hidden unit in all 50 trials while it used at least one hidden unit for the remaining eight months. Especially for the month of December the network used as many as 17 hidden units on a few trials. This suggests that the networks developed by the NN1 model are capable of adopting their complexity to the complexity of the data.

The NN2 model always used a significantly smaller number of hidden units than the NN1 model. In experiment 1, the network learned without a hidden unit in 33 trials and with one hidden unit in the remaining 17 trials. It should be noted that networks with no hidden units are linear models. In experiment 2, the network managed to learn the flow history without a hidden unit in 39 trials, with 1 hidden unit in 10 trials and with 2 hidden units in the remaining trial. In experiment 3, the NN2 model learned the flow history without a hidden unit for the months of February and April through July and with a variable number of hidden units for the remaining months. As in the NN1 model, the NN2 model also used maximum number of hidden units for the month of December. One reason why the NN2 model needed fewer hidden units than the NN1 model may be due to the fact that it had inputs with explicit time-lag information, whereas, the NN1 model had inputs that did not have sufficient time-lag information. This also suggests that the time-delay window used in the NN2 model might have helped the network to develop a more appropriate smoothing than the five-day averaging.

The connection weights of a trained network are kept frozen during testing. This means that we can express the actual computation performed by a network in terms of an equivalent mathematical expression. Such expressions can be helpful for further understanding and analysis. In what follows next we give equivalent expressions for both the NN1 and NN2 models. Since it may not be possible to enumerate expressions for all possible networks we restrict ourselves to those networks that used zero, one or two hidden units. (Note that for the networks with 3 or more hidden units, it is quite straightforward to extend the equations given below.) The expressions developed below

are only for $net \geq 0$ because we are interested only in positive outputs.

For the NN1 model without a hidden unit, the network is equivalent to a simple linear model of the form

$$\hat{\mathrm{D}}(\mathrm{t}) = \mathbf{W}_{\mathrm{I,O}}\mathbf{I}(\mathrm{t}) \tag{11}$$

where $\hat{\mathrm{D}}(\mathrm{t})$ is the flow at the Dexter gage at time $t$ and $\mathbf{W}_{\mathrm{I,O}} = [\mathrm{w}_{\mathrm{b,o}}, \mathrm{w}_{\mathrm{H,o}}, \mathrm{w}_{\mathrm{M,o}}, \mathrm{w}_{\mathrm{A,o}}]$ is a weight vector that represents the connections from the input layer to the output unit. Here $\mathrm{w}_{\mathrm{b,o}}$ represents the weight of the connection from a special unit called the bias unit whose output is always 1. The remaining three weights ($\mathrm{w}_{\mathrm{H,o}}, \mathrm{w}_{\mathrm{M,o}}, \mathrm{w}_{\mathrm{A,o}}$) correspond to connections from the Hamburg, Mill Creek and Ann Arbor gauges respectively. The column vector $\mathbf{I}(\mathrm{t}) = [1, \mathrm{H}(\mathrm{t}), \mathrm{M}(\mathrm{t}), \mathrm{A}(\mathrm{t})]^{\mathrm{T}}$ is a transpose of the input vector at time $t$.

For a network with 1 hidden unit the equivalent nonlinear model is given by the following expression

$$\hat{\mathrm{D}}(\mathrm{t}) = \mathbf{W}_{\mathrm{I,O}}\mathbf{I}(\mathrm{t}) + \mathbf{W}_{\mathrm{H,O}}\mathbf{HU}(\mathrm{t}) \tag{12}$$

The first term in equation 12 represents a linear model similar to that of a network with zero hidden unit (equation 11). The weight vector $\mathbf{W}_{\mathrm{H,O}} = [\mathrm{w}_{\mathrm{h1,o}}]$ in the second term in the equation 12 represents the weight of the connection from the hidden unit ($h1$) to the output unit. The activation vector $\mathbf{HU(t)} = [\mathrm{h}_1(\mathrm{t})]^{\mathrm{T}}$ in the above expression represents the output of the hidden unit. Since the hidden unit is a sigmoidal unit, the activation of the hidden unit is given by

$$\mathrm{h}_1(\mathrm{t}) = \frac{1}{1 + e^{-(\mathbf{W}_{\mathrm{I,h1}}\mathbf{I}(\mathrm{t}))}} \tag{13}$$

where $\mathbf{W}_{\mathrm{I,h1}}$ is the weight vector for the connections from the input units to the hidden unit $h1$. Thus the second term of the equation 12 represents the nonlinear component of the model. Note that this network has 9 weighted connections (5 input connections of the output unit and 4 inputs connections of the hidden unit).

Similarly for a network with 2 hidden units the resulting expression is

$$\hat{\mathrm{D}}(\mathrm{t}) = \mathbf{W}_{\mathrm{I,O}}\mathbf{I}(\mathrm{t}) + \mathbf{W}_{\mathrm{H,O}}\mathbf{HU}(\mathrm{t}) \tag{14}$$

where $\mathbf{W}_{\mathrm{H,O}} = [\mathrm{w_{h1,o}}, \mathrm{w_{h2,o}}]$ is a two component weight vector from the hidden units $h1$ and $h2$, and $\mathbf{HU}(\mathrm{t}) = [\mathrm{h_1(t)}, \mathrm{h_2(t)}]^{\mathrm{T}}$ is the activation vector of the hidden units. The activation of the first hidden unit $\mathrm{h_1(t)}$ is same as that of a network with 1 hidden unit (equation 13), while the activation of the second hidden unit is a second order sigmoidal expression,

$$\mathrm{h_2(t)} = \frac{1}{1 + e^{-(\mathbf{W}_{\mathrm{I,h2}}I(t) + \mathrm{w_{h1,h2}h_1(t)})}} \tag{15}$$

Note that the hidden unit h2 receives input not only from the input units and the bias unit but also from the first hidden unit (via the weight $\mathrm{w_{h1,h2}}$).

By following the above steps, we can write similar expressions for the NN2 model. Since the NN2 model has a five-day window for each of the three inputs the corresponding input and weight vectors should be modified accordingly. Thus for the NN2 model the modified weight vector for the connections from the input layer to the output unit is

$$\mathbf{W}_{\mathrm{I,O}} = [\mathrm{w_{b,o}}, \mathrm{w_{H(t-4),o}}, \cdots, \mathrm{w_{H(t),o}}, \mathrm{w_{M(t-4),o}}, \cdots, \mathrm{w_{M(t),o}}, \mathrm{w_{A(t-4),o}}, \cdots, \mathrm{w_{A(t),o}}].$$

The modified input vector with a five-day window is given by

$$\mathbf{I}(\mathrm{t}) = [1, \mathrm{H(t-4)}, \cdots, \mathrm{H(t)}, \mathrm{M(t-4)}, \cdots, \mathrm{M(t)}, \mathrm{A(t-4)}, \cdots, \mathrm{A(t)}]^{\mathrm{T}}.$$

Similarly the weight vectors $\mathbf{W}_{\mathrm{I,h1}}$ and $\mathbf{W}_{\mathrm{I,h2}}$ feeding the hidden units $h1$ and $h2$ can be rewritten to accommodate the additional inputs. However, vectors $\mathbf{HU}(\mathrm{t})$ and $\mathbf{W}_{\mathrm{H,O}}$ need not be modified. By substituting these modified vectors in equations 11 through 15 we can obtain equivalent expressions for the NN2 model with zero, 1, or 2 hidden units.

The above equations imply that the resulting models (i.e., networks) become progressively complex with the addition of each hidden unit. Thus, the neural network approach

is able to develop complex models using appropriate degrees of linear and nonliner components to match the complexity of the flow process represented by the training set. This flexibility is not available in the power model.

### 4.2 Computational Requirement

The Cascade-Correlation algorithm is a very efficient algorithm. It trained the NN2 networks within a few hundred iterations and the NN1 networks within a few hundred to a few thousand iterations depending on the number of hidden units installed. However, from the computational point of view, the neural network approach may seem to be less advantageous over solving the power model using a spreadsheet program, because the user may have to train the network several times to reduce the variations in the prediction results. But, such a direct comparison between a spreadsheet program and developing neural network models with the Cascade-Correlation algorithm may not be meaningful for the following reasons: 1) the neural network approach used here is a "black box" approach (i.e., the user need not spend time in searching for an appropriate analytic model for the flow process), and 2) there is no need for transforming the data (for example, using a logarithmic transformation) to fit a model to the data.

## 5 Conclusions

This paper evaluated the applicability of neural networks with a clipped linear output unit for river flow prediction and presented some preliminary results. The $mse$ values for the testing period suggest that, among the two neural network models considered, the NN2 model seems to be a more accurate predictor than the power model. This suggests that representing the input using a five-day window is more useful than the five-day averages. A comparison of $re$ values indicate that the neural networks are capable of

predicting the high runoffs better than the power model. Our results also suggest that, unlike the power model, the neural network models are less affected by the data on the Mill Creek tributary and that it may not be necessary for the user to do any smoothing or transformation on the input data.

The neural network approach is a "black box" approach and the user need not know much about the flow process. The Cascade-Correlation algorithm makes the applicability of the neural network approach quite easy because it relieves the user from concentrating on the issues such as convergence of the learning algorithm and the selection of a suitable architecture for the network. Our analysis on the size of the networks suggests that the Cascade-Correlation algorithm is capable of adapting the complexity of the neural networks to match the complexity of the training set. This flexibility is not available in analytic model such as the power model because the complexity of the model is fixed a priori.

However, we do note that the neural network approach is a new approach for this application and further research is needed to fully understand its modeling capability. Hence, we plan to further investigate the applicability of the neural network approach in the following directions: 1) applying these neural network models for predicting other missing flow histories, 2) comparing the predictive accuracy of our neural network models with parametric models, and 3) exploring the applicability of recurrent networks.

**Acknowledgements**

The authors would like to thank Dr. Scott Fahlman for providing the initial code for his algorithm. We sincerely appreciate the reviewers for their valuable comments and efforts.

## Appendix I: References


Borland (1991). "QuattroPro Version 4.0, Users Guide", *Borland International*, Scotts Valley, CA.

Beven, K. (1989). "Changing Ideas in Hydrology-The Case of Physically-Based Models". *J. Hydrology*, 105, 157-172.

Bovee, K. D. (1992). "Relations Between Streamflow, Habitat and Populations of Smallmouth Baas and Rock Bass in the Huron River, Michigan". *U.S. Fish and Wildelife Service, Biological Report*, 1992.

Cheu, R. L., Ritchie, S. G., Recker, W. W., and Bavarian, B. (1991). "Investigation of a Neural Network Model for Freeway Incident Detection". *Proc. 2nd Int. Conf. on the Applicability of AI to Civil and Structural Eng.*. B. H. V. Topping, ed., Vol. 1, AI and Civil Eng., Oxford, England. Civil-Comp Press, pp. 267-274.

Chow, D. (1964). *Handbook of Applied Hydrology.* McCraw-Hill Book Company, New York, Sec. 15.

Fahlman, S. E., and Lebiere, C. (1990). "The Cascaded-Correlation Learning Architecture". School of Comput. Sci., Carnegie Mellone University, Pittsburg, Tech. Rep. CMU-CS-90-100, Feb. 1990.

Furuta, H., Sugiura, K., Tonegawa, T., and Wanatabe, E. (1991). "A Neural Network System for Aesthetic Design of Dam Structures". *Proc. 2nd Int. Conf. on the Applicability of AI to Civil and Structural Eng.*. B. H. V. Topping, ed., Vol. 2, AI and Structural Eng., Oxford, England. Civil-Comp Press, pp. 273-278.

Crawford, N. H., and Linsley, K. K. (1966). "Digital Simulation of Watershed: Stanford Watershed Model-IV". Civil Engineering Dept., Stanford University, Tech. Rep. 39, 1966.

Hajela, B., Fu, B., and Berke, L. (1991). "ART Networks in Automated Conceptual Design of Structural Systems". *Proc. 2nd Int. Conf. on the Applicability of AI to Civil and Structural Eng.*. B. H. V. Topping, ed., Vol. 2, AI and Structural Eng., Oxford, England. Civil-Comp

Press, pp. 263-271.

Kamarthi, S. V., Sanvido, V. E., and Kumara, S. R. T. (1992). "Neuroform-Neural Network System for Vertical Formwork Selection". *J. Comput. Civ. Eng.*, ASCE, 6(6), 178-199.

Karunanithi, N., Whitley, D., and Malaiya, Y. K. (1992a). "Prediction of Software Reliability Using Connectionist Models". *IEEE Trans. on Software Eng.*, 18(7), July 1992, 563-574.

Karunanithi, N., Whitley, D., and Malaiya, Y. K. (1992b). "Using Neural Networks in Reliability Prediction". *IEEE Software*, 9(4), July 1992, 53-59.

Karunanithi, N. (1992c). "Generalization in the Cascade-Correlation Architecture: Some Experiments and Applications". *Ph.D. Dissertation*, Computer Science Dept., Colorado State University, Fort Collins, CO, Fall 1992.

Lippmann, R. P. (1987). "An Introduction to Computing with Neural Nets". *IEEE ASSP Mag.*, 4(2), 4-22.

McCuen, R. H., Leahy, R. B., and Johnson, P. A. (1990). "Problems with Logarithmic Transformations in Regression". *Journal of Hydrolic Eng.*, ASCE, 116(3), 414-428.

Moselhi, O., Hagazy, T., and Fazio, P. (1991). "Markup Estimation Using AI Methodology". *Proc. 2nd Int. Conf. on the Applicability of AI to Civil and Structural Eng.*. B. H. V. Topping, ed., Vol. 1, AI and Civil Eng., Oxford, England. Civil-Comp Press, pp. 257-266.

Ramirez, M. R., and Arghya, D. (1991). "A Faster Learning Algorithm for BP Neural Networks in NDE Application". *Proc. 2nd Int. Conf. on the Applicability of AI to Civil and Structural Eng.*. B. H. V. Topping, ed., Vol. 1, AI and Civil Eng., Oxford, England. Civil-Comp Press, pp. 275-283.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1987). "Learning Internal Representations by Error Propagation." *Parallel Distributed Processing: Explorations in the microstructure of cognition*, D. E. Rumelhart and J. L. McClelland, eds., Vol. 1, MIT Press, Cambridge, Mass, 318-362.

Weigend, A. S., Huberman, B. A., and Rumelhart, D. E. (1990). "Predicting the Future: A Connectionist Approach". *Int. Journal of Neural Systems.*, 1, 193-209.

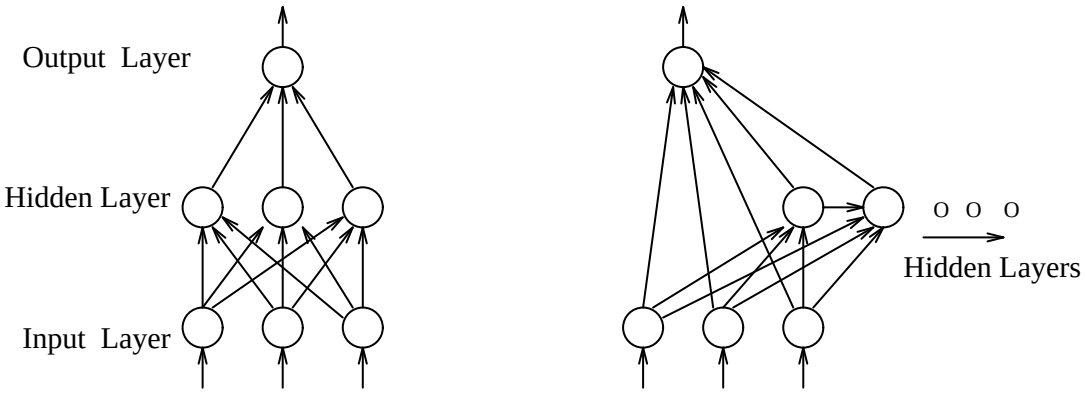
Whitley, D., and Karunanithi, N. (1991). "Generalization in Feedforward Neural Networks". *Proc. Int. Joint Conf. on Neural Networks, Seattle, Washington.*, Vol. II, 77-82, July 1991, IEEE Press.

Xihui, L., Baocheng, S., and Wenyuan, F. (1991). "Fuzzy Reasoning with Engineering Applications using Neural Networks". *Proc. 2nd Int. Conf. on the Applicability of AI to Civil and Structural Eng.*. B. H. V. Topping, ed., Vol. 2, AI and Structural Eng., Oxford, England. Civil-Comp Press, pp. 279-285.

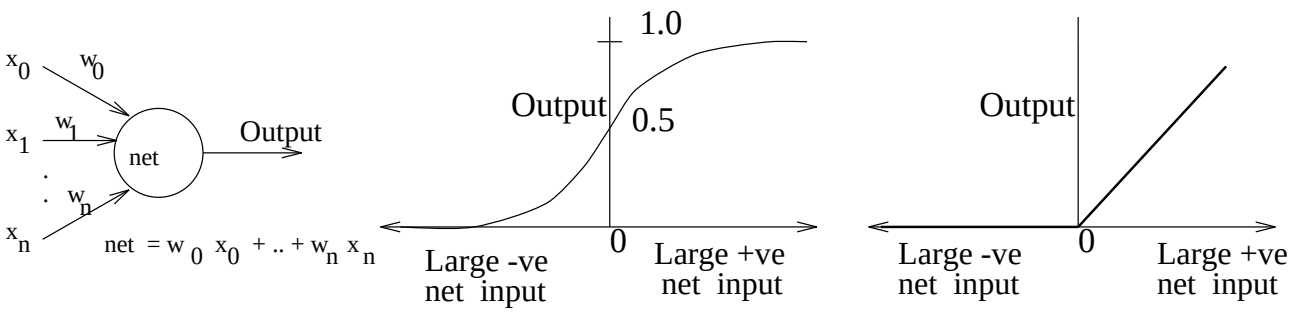
## Appendix II: Notations

The following symbols are used in this paper:

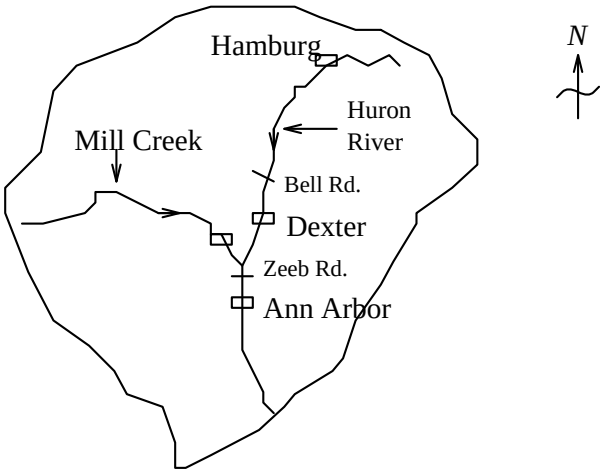
| | | |
|---|---|---|
| A(t) | = | discharge measured at Ann Arbor at time $t$; |
| b | = | label of the bias unit; |
| D(t) | = | discharge measured at Dexter at time $t$; |
| $\widehat{\mathrm{D}}(t)$ | = | predicted flow at Dexter at time $t$; |
| $\mathrm{D}_i$ | = | actual observed flow at Dexter corresponding to the input $i$; |
| $\widehat{\mathrm{D}}_i$ | = | predicted flow at Dexter for the input $i$; |
| $\mathbf{HU(t)}$ | = | activation vector of hidden units at time $t$; |
| H(t) | = | discharge measured at Hamburg at time $t$; |
| hi | = | label of the hidden unit $i$; |
| $\mathrm{h_i(t)}$ | = | output activation of hidden unit $i$ at time $t$; |
| $\mathbf{I}$(t) | = | transpose of the input to the neural network at time $t$; |
| M(t) | = | discharge measured at Mill Creek at time $t$; |
| $n$ | = | number of points used in the test set; |
| NN1 | = | 3-input, 1-output neural network; |
| NN2 | = | 15-input, 1-output neural network; |
| o | = | label of the output unit; |
| $\bar{re}$ | = | mean of the percentage relative prediction error; |
| $sse$ | = | sum square error; |
| net | = | weighted sum of inputs to a neuron; |
| t | = | time index of the current point; |
| (t − i) | = | time index of the $i_{th}$ point prior to the current point; |
| $\mathbf{W}_{\mathrm{I,O}}$ | = | weight vector from the input layer to the output unit; |
| $\mathbf{W}_{\mathrm{H,O}}$ | = | weight vector from hidden units to the output unit; |
| $\mathbf{W}_{\mathrm{I,hi}}$ | = | weight vector from the input layer to the hidden unit $hi$; |
| $\mathrm{w_{i,j}}$ | = | weight from unit $i$ to unit $j$; and |
| $\beta_0, \beta_1, \beta_2$ | = | parameters of the power model. |

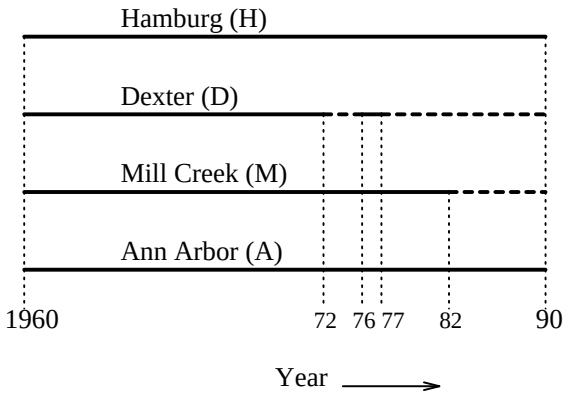


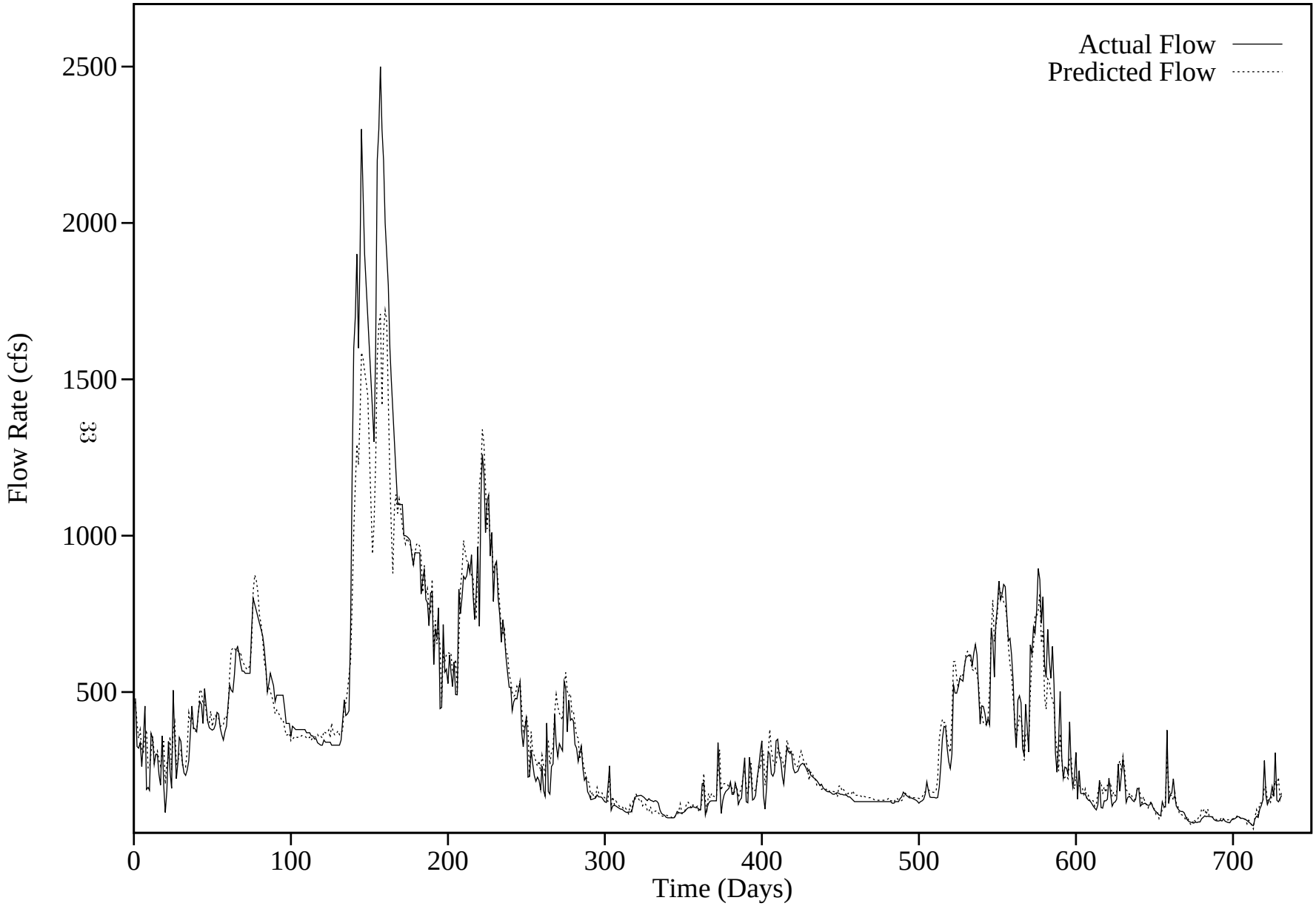

(a) A Typical BP-network (b) A Typical CASCADE-network

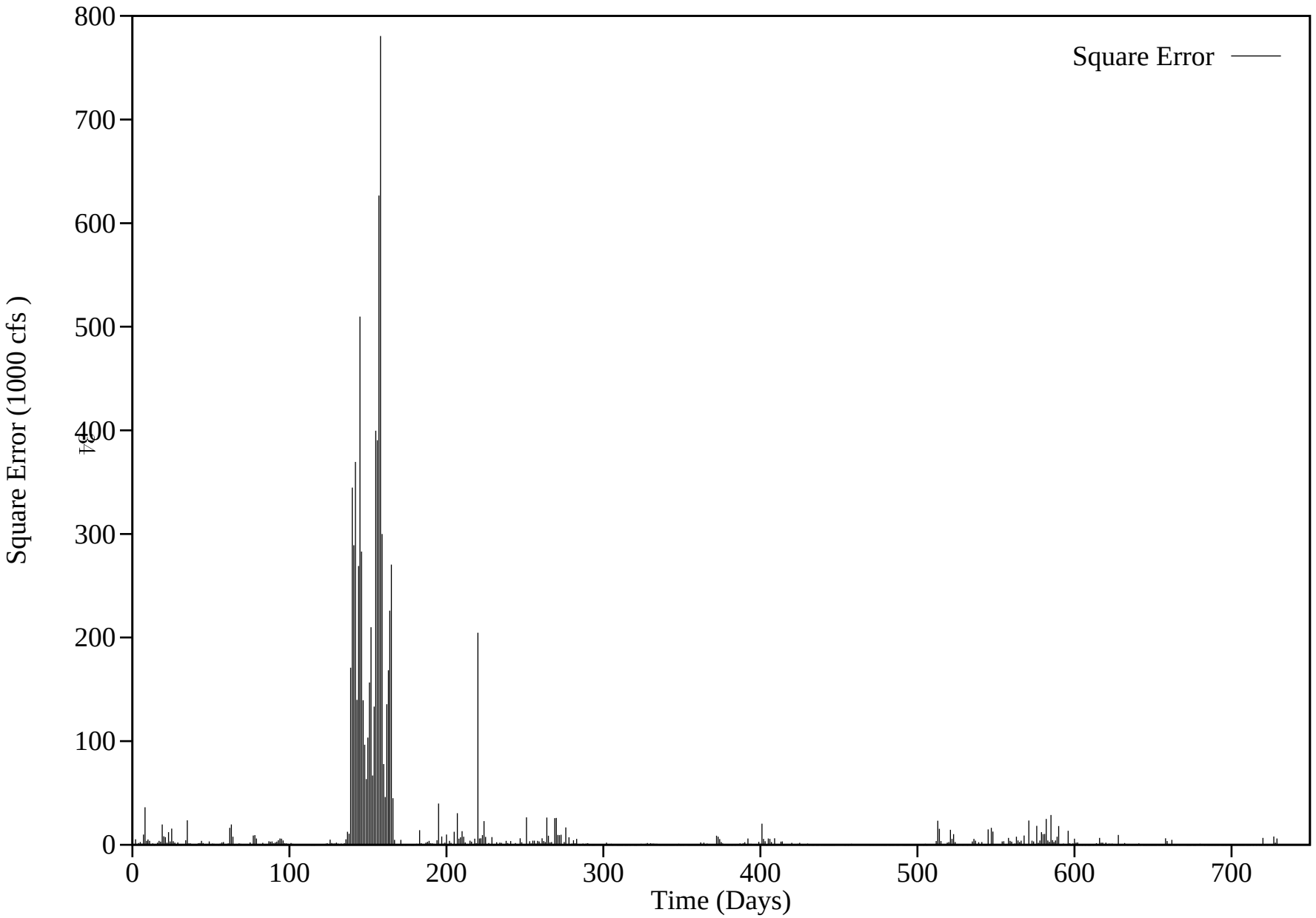


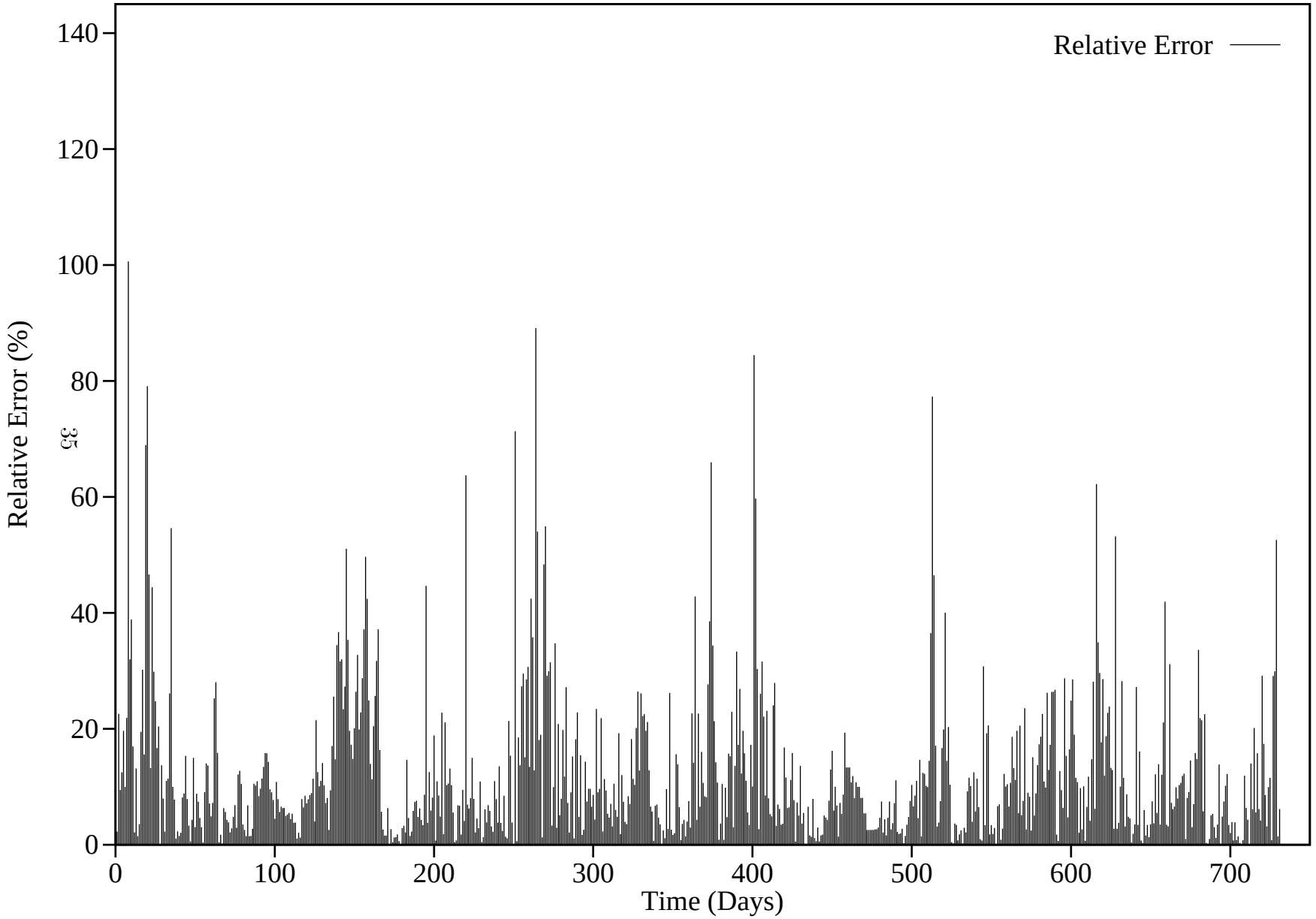

( a ) A Typical Processing Unit ( b ) Logistic Function (c) Linear Function

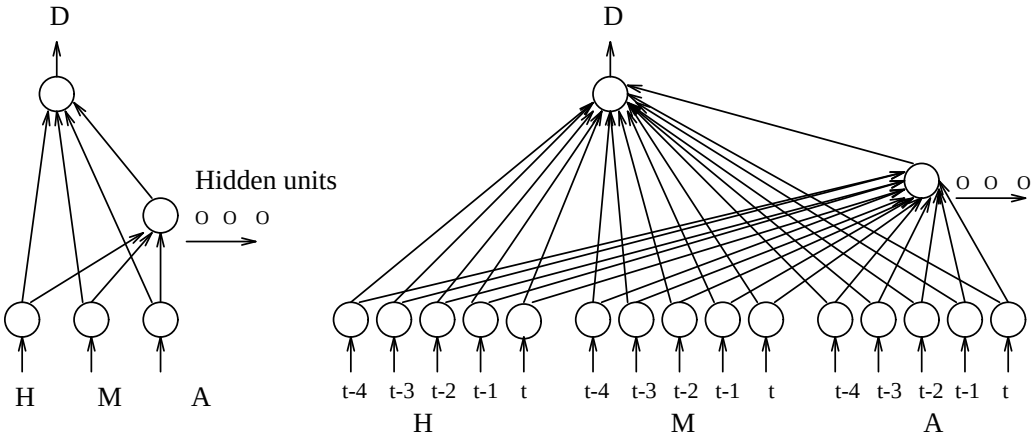

Hamburg
N
Huron
River
Mill Creek
Bell Rd.
Dexter
Zeeb Rd.
Ann Arbor


Hamburg (H)
Dexter (D)
Mill Creek (M)
Ann Arbor (A)
1960
72
76
77
82
90
Year

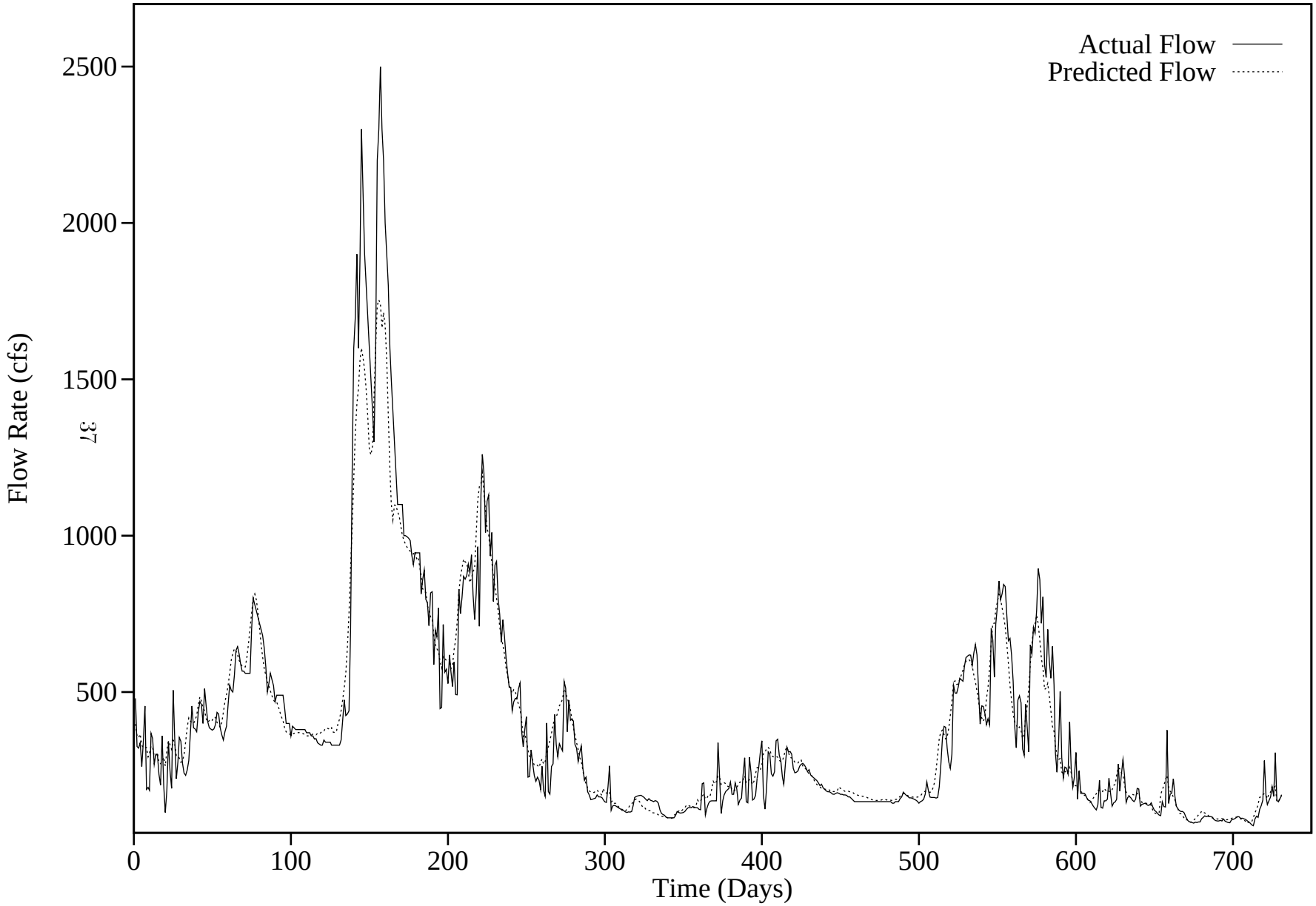

Actual Flow
Predicted Flow
2500
2000
1500
1000
500
Flow Rate (cfs)
0
100
200
300
400
500
600
700
Time (Days)

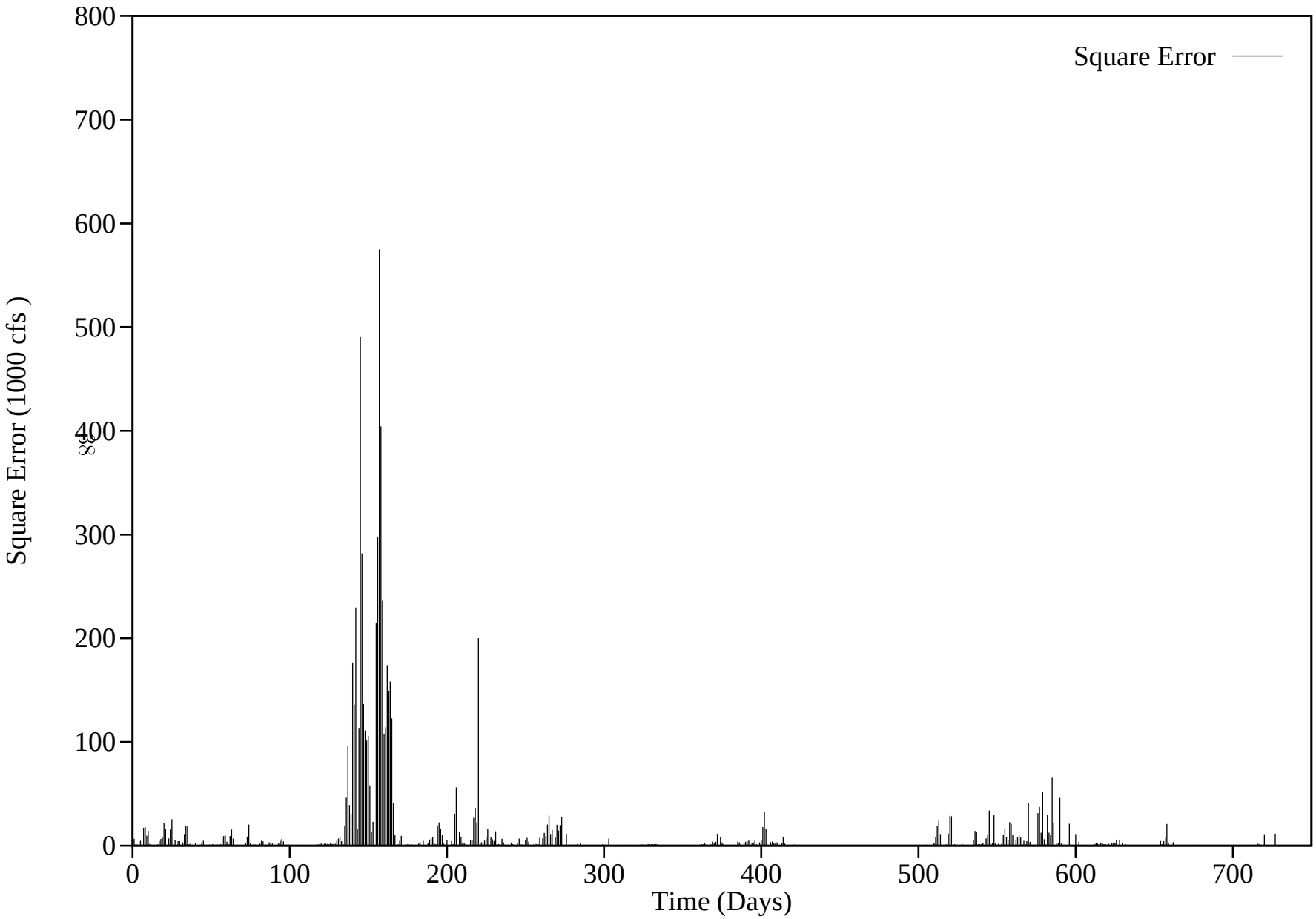

Square Error
Square Error (1000 cfs )
Time (Days)
0
100
200
300
400
500
600
700
800


Relative Error
Relative Error (%)
ε
0
20
40
60
80
100
120
140
0
100
200
300
400
500
600
700
Time (Days)




(a) A Standard CASCADE-network (b) A CASCADE-network with a Five-day Window


Actual Flow
Predicted Flow
2500
2000
1500
1000
500
Flow Rate (cfs)
0
100
200
300
400
500
600
700
Time (Days)


Square Error
Square Error (1000 cfs )
Time (Days)
0
100
200
300
400
500
600
700
800

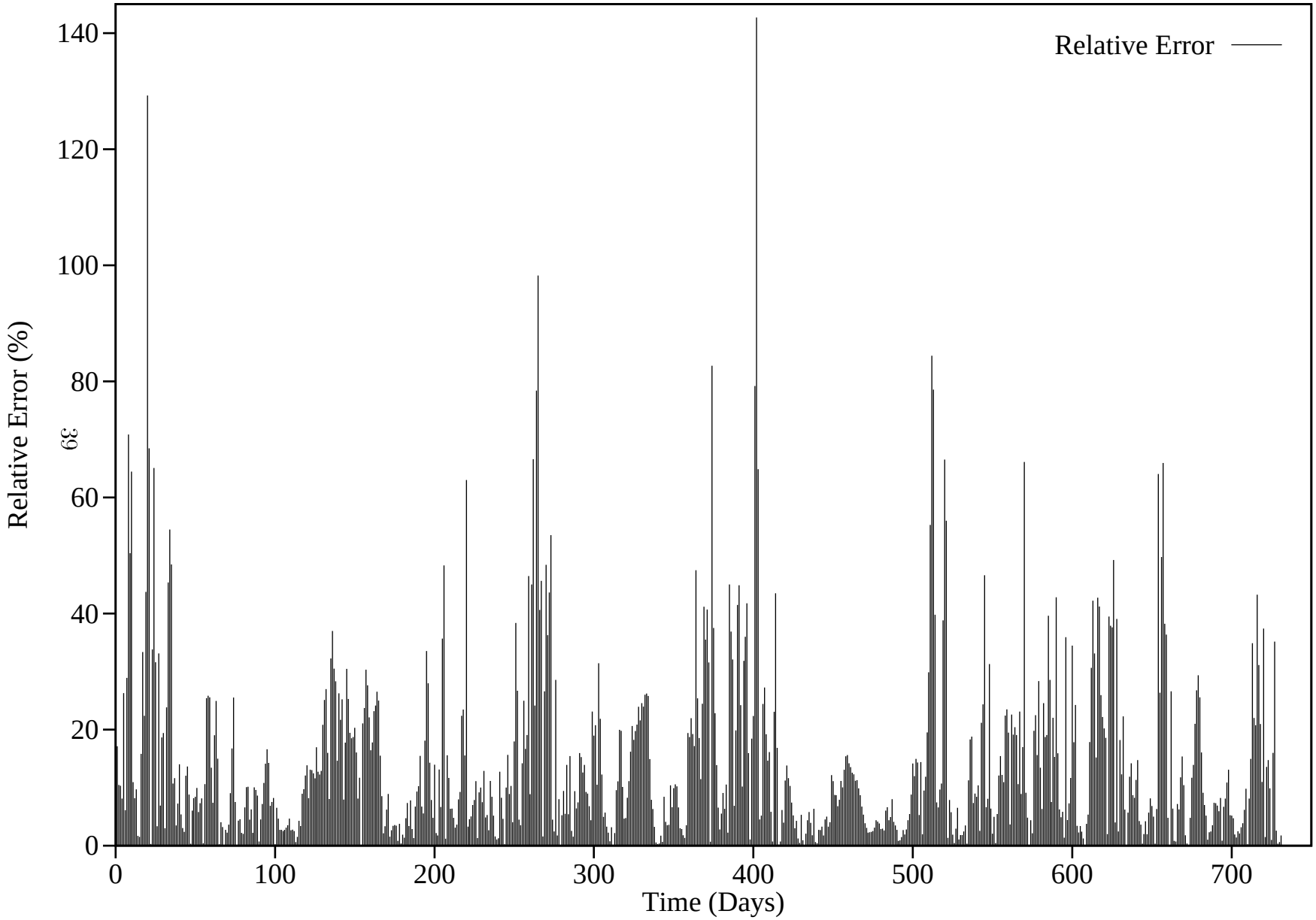

Relative Error
Relative Error (%)
Time (Days)
0
20
40
60
80
100
120
140
0
100
200
300
400
500
600
700

Table 1: **Testing and Training Errors of Neural Networks and the Power Model.**

| Experiment Type | Power Model | | NN1 | | NN2 | |
|---|---|---|---|---|---|---|
| | *mse* | *mre* | *mse* | *mre* | *mse* | *mre* |
| Testing Errors | | | | | | |
| 1 | 11,763 | 11.9 | 11,365 | 12.4 | 7,044 | 8.0 |
| 2 | 6,372 | 7.5 | 7,272 | 7.6 | 1,522 | 8.3 |
| 3 | 12,598 | 8.5 | 10,221 | 8.9 | 10,104 | 9.5 |
| Training Errors | | | | | | |
| 1 | 2,782 | 12.3 | 1,800 | 11.2 | 991 | 8.5 |
| 2 | 1,415 | 8.4 | 995 | 8.0 | 249 | 8.5 |
| 3 | 952 | 7.1 | 681 | 7.6 | 670 | 7.9 |

Table 2: **A Summary of Hidden Units Used in Neural Networks**

| Experiment Type | NN1 | | | | NN2 | | | |
|---|---|---|---|---|---|---|---|---|
| | Average | Min | Max | SD | Average | Min | Max | SD |
| 1 | 15.92 | 12 | 21 | 2.07 | 0.34 | 0 | 1 | 0.48 |
| 2 | 1.16 | 1 | 2 | 0.37 | 0.24 | 0 | 2 | 0.48 |
| 3 | 2.53 | 0 | 17 | 3.45 | 0.58 | 0 | 5 | 0.97 |

**Key Words**

River Flow Prediction, Model Evaluation, Power Model, Neural Networks, Adaptive Model Synthesize, Cascade-Correlation Algorithm, Complex Models.

**Figure Captions**

Figure 1: A Typical BP-network and a CASCADE-network.

Figure 2: A Typical Processing Unit Used in the Output and Hidden Layers of a Neural Network.

Figure 3: The Huron River and the Mill Creek Tributary.

Figure 4: Flow Records Used in This Study.

Figure 5: Observed Vs. Predicted Flows at Dexter by the Power Model for the Testing Period 1976 and 1977.

Figure 6: Square Error of the Power Model for Flows at Dexter for the Testing Period 1976 and 1977.

Figure 7: Relative Error of the Power Model for Flows at Dexter for the Testing Period 1976 and 1977.

Figure 8: Neural Network Models Used for Prediction.

Figure 9: Observed Vs. Predicted Flows at Dexter by the NN1 Model for the Testing Period 1976 and 1977.

Figure 10: Square Error of the NN1 Model for Flows at Dexter for the Testing Period 1976 and 1977.

Figure 11: Relative Error of the NN1 Model for Flows at Dexter for the Testing Period 1976 and 1977.