Hydrol. Earth Syst. Sci., 16, 1151–1169, 2012
www.hydrol-earth-syst-sci.net/16/1151/2012/
doi:10.5194/hess-16-1151-2012

Hydrology and Earth System Sciences

Dynamic versus static neural network model for rainfall forecasting at Klang River Basin, Malaysia

A. El-Shafie[1], **A. Noureldin**[2], **M. Taha**[1], **A. Hussain**[3], **and M. Mukhlisin**[1]

[1]Civil and Structural Engineering Dept. University Kebangsaan Malaysia, Malaysia
[2]Electrical and Computer Engineering, Royal Military College, Kingston, Canada
[3]Electric, Electronics Systems Engineering Dept. University Kebangsaan Malaysia, Malaysia

Correspondence to: A. El-Shafie (elshafie@vlsi.eng.ukm.my)

Received: 3 June 2011 – Published in Hydrol. Earth Syst. Sci. Discuss.: 6 July 2011
Revised: 26 March 2012 – Accepted: 27 March 2012 – Published: 10 April 2012

Abstract. Rainfall is considered as one of the major components of the hydrological process; it takes significant part in evaluating drought and flooding events. Therefore, it is important to have an accurate model for rainfall forecasting. Recently, several data-driven modeling approaches have been investigated to perform such forecasting tasks as multi-layer perceptron neural networks (MLP-NN). In fact, the rainfall time series modeling involves an important temporal dimension. On the other hand, the classical MLP-NN is a static and has a memoryless network architecture that is effective for complex nonlinear static mapping. This research focuses on investigating the potential of introducing a neural network that could address the temporal relationships of the rainfall series.

Two different static neural networks and one dynamic neural network, namely the multi-layer perceptron neural network (MLP-NN), radial basis function neural network (RBFNN) and input delay neural network (IDNN), respectively, have been examined in this study. Those models had been developed for the two time horizons for monthly and weekly rainfall forecasting at Klang River, Malaysia. Data collected over 12 yr (1997–2008) on a weekly basis and 22 yr (1987–2008) on a monthly basis were used to develop and examine the performance of the proposed models. Comprehensive comparison analyses were carried out to evaluate the performance of the proposed static and dynamic neural networks. Results showed that the MLP-NN neural network model is able to follow trends of the actual rainfall, however, not very accurately. RBFNN model achieved better accuracy than the MLP-NN model. Moreover, the forecasting accuracy of the IDNN model was better than that of static network during both training and testing stages, which proves a consistent level of accuracy with seen and unseen data.

1 Introduction

2 Background

Characteristics and amount of rainfall are not easily known until it occurs. As rainfall plays a crucial role in evaluation and management of drought and flood events, it is very important to be able to forecast rainfall.

Developing optimal release policies of multi-purpose reservoirs is very complex, especially for reservoirs with an explicit stochastic environment (e.g. uncertainty in future rainfall that reflects the amount of reservoir inflow). The development of management models for identification of optimal operating policies for reservoirs spans over four decades of research. In a random environment, where climatic factors such as stream flow are stochastic, the economic returns from reservoir releases defined by the optimal policy are uncertain. Furthermore, the consequences of release decision cannot be fully realized until future unknown (rainfall/inflow) events occur. As a result, rainfall forecasts become an essential requirement for Klang Gate Reservoir operation. Accurate forecasting means better control of water availability and more refined operation of reservoirs. Therefore, the problem of forecasting rainfall at Klang River is of considerable importance for better water management at Klang Gate Reservoir.

However, in the past most of the methods used in rainfall forecasting are regression or auto-regression linear models and their ability is limited in dealing with natural phenomenon with a non-linear trend (de Vos and Rientjes, 2005; Hung et al., 2009; Modarres, 2009). Time variations of rainfall rate have always been forecasted for actual use in advance of the daily activities. It is important to mention that models for rainfall forecasting are fundamental tools in water resources studies, since they determine and provide the basis in establishing future reservoir water inflows (Akhtar et al., 2009; Anctil and Lauzon, 2004; Awwad et al., 1994). These predictions are of significant importance in the planning of water resources systems, being responsible for the optimization of the system as a whole. That is why rainfall forecasting is a fundamental topic in many engineering applications like constructing dams, analysis and forecasting, planning and designing of reservoirs, hydro-power generation, irrigation, water management, controlling floods and others.

The rainfall forecasting problem has been traditionally tackled using linear techniques, such as AR, ARMAX, and Kalman filter, but also using nonlinear regression (see Cheng, 1994; Alvisi et al., 2006; Abrahart and See, 2007; Bras and Rodriguez-Iturbe, 1985; Chiu, 1978; Box and Jenkins, 1970). Most of the forecasting methods consider one day ahead forecast. For the rainfall a longer term forecast such as ten days ahead or a month ahead is more of interest, though it is more difficult than the one day ahead problem. In fact, there are several considerable drawbacks to the use of KF in rainfall forecasting application. These include (1) the necessity of accurate stochastic modeling, which may not be possible in the case for rainfall; (2) the requirement for a priori information of the system measurement and develop covariance matrices for each new pattern, which could be challenging to accurately determine and (3) the weak observability of some of temporal pattern states that may lead to unstable estimates for the forecasted value (see Noureldin, et al., 2007, 2011).

In this context, motivation for utilizing non-linear modeling approach based on the Artificial Intelligence (AI) techniques has received considerable attention from the hydrologists in the last two decades (Boucher et al., 2010; de Vos and Rientjes, 2005; Toth et al., 2000; Weigend et al., 1995; Xiong et al., 2004). Lapedes and Farber (1987) conducted a study on detection of nonlinear response and damage detection on signal processing, and concluded that ANN can be used for modeling and forecasting nonlinear time series. Recently, numerous ANN-based rainfall-runoff models have been proposed to forecast streamflow (Hsu et al., 1995; Thirumalaiah and Deo, 1998, 2000; Campolo et al., 1999; Sajikumar and Thandaveswara, 1999; Tokar and Johnson, 1999; Zealand et al., 1999) and reservoir inflow (Saad et al., 1996; Jain et al., 1999; Coulibaly et al., 2000a, b). In addition, neural networks and fuzzy logic have been used as effective modeling tools in different environmental processes such as wastewater treatment, water treatment and air pollution. Several water quality prediction models have been developed utilizing ANN and ANFIS methods (Najah et al., 2010a, b, 2011). Rainfall-runoff models utilizing ANN model showed significant level accuracy if compared with traditional regression models (El-Shafie et al., 2011a). Cinar et al. (2006) used an artificial neural network to predict the performance of a membrane bioreactor. They were able to estimate concentrations of chemical oxygen demand, phosphate, ammonia and nitrate. Altunkaynak et al. (2005a) used fuzzy logic modeling to forecast dissolved oxygen concentration. Altunkaynak et al. (2005b) compared the accuracy of fuzzy logic modeling and autoregressive integrated moving average (ARIMA) models in predicting water consumption in a city. They found that relative error rates for fuzzy logic and ARIMA were 2.5 and 2.8, respectively.

Recently, the authors developed several AI-based inflow forecasting architectures using multi-layer perceptron neural networks (MLPNN), radial basis function neural networks (RBFNN) and adaptive neuron-fuzzy inference systems (ANFIS) at Aswan High Dam, Nile River, Egypt (Elshafie and Noureldin, 2011; El-Shafie et al., 2011b, c). The main idea behind all of these methods is to mimic the latest inflow pattern to forecast the inflow for 3 months ahead. The major drawback of such models is the lack of ability to automatically mimic the temporal inflow pattern trend during the model training stage procedure. Therefore, any of the mentioned AI-based models may not be capable of providing a reliable and accurate forecasting solution.

2.1 Problem statement

In light of the above literature, it could be concluded that previous studies on rainfall forecasting models face two major deficiencies. The first is linearization feature in the modeling methods, whether in KF and ARMA model, while the rainfall behavior is a non-linear sequence in nature. Such deficiency was elucidated by the motivation of the AI-based models; thus, the non-linearity feature of input/output mapping could be examined. The second is the temporal feature of the rainfall pattern, which is not completely considered in many classical AI-based models. In fact, in static neural networks the output is directly calculated from the input through forward connections. In this context, for the second deficiency, in order to have an accurate rainfall forecasting model, it is necessary to utilize a modeling approach that could consider the temporal feature in the rainfall pattern. Dynamic neural networks include delay lines between the layers. So the output depends also on previous inputs and/or previous states of the network. The dynamic neural networks in which all layers have feedback connections with several time delays mean that the temporal feature could be considered in the model structure. More details about the structure of the dynamic neural network will be reported in Sect. 2.2.

2.2 Objective

In this research, we aim at developing an AI-based rainfall forecasting model, taking into consideration the extensive temporal pattern trend and thus providing a better forecasting accuracy. Such a technique combines the advantages of some of the existing models with the advantages of dynamic neural networks in representing the sequential process in the input data (the latest rainfall pattern). In this way, it should be possible for the proposed model to mimic the temporal pattern of the rainfall, based on the current and some past patterns. The proposed model will be tested utilizing real rainfall records at Klang River, Malaysia. Finally, comprehensive comparative analyses are performed in order to examine the significance of utilizing the dynamic neural network over the classical static neural network methods.

3 Rainfall forecasting model

Let us consider a simple forecasting model that attempts to issue a prediction $R(t+T)$ of the rainfall in a specified week/month, based on explanatory variables, say the historical rainfall records of previous weeks/months with a certain length $kR(t+T-k)$. Given a time series of data $R(t)$, the auto regression model is a tool for understanding and predicting future values in this series. The model is usually then referred to as below:

$$R(t+T) = f\left(R(t), R(t-1), ..., R(t-k)\right) \quad (1)$$

where R is the rainfall, $T=1$, and $k=w$ (where w is the input window length; the selection of the window size will be described later in Sect. 3). The model basically says that the rainfall next week (month) is some combination (average) of the rainfalls observed now and one – two weeks (months) ago.

3.1 Autoregressive-moving-average (ARMA)

The ARMA model was developed utilizing the same rainfall data at Klang River. These data were analyzed in the time-domain while fitting a model of the following form:

$$y_t = \phi_1 y_{t-1} + ... + \phi_p y_{t-p} + \theta_1 \varepsilon_{t-1} + ... \phi_q \varepsilon_{t-q} + \varepsilon_t + \delta, \quad (2)$$

which best predicts the values of variable Y at time t based on previous observations, $y_{t-1}...y_{t-p}$, previous error terms $\varepsilon_{t-1}... \ \varepsilon_{t-q}$, and a constant, δ. The q values are collectively referred to as the autoregressive part of the model (of order p), whereas θs constitute the moving-average component (of order q). The inclusion of a nonzero δ introduces a deterministic trend in the model. We refer to this stochastic process as an autoregressive-moving-average model (ARMA(p,q)), and we are concerned with identifying the order of the model and estimating its coefficients. Models in which there are no moving-average terms (i.e. $q=0$) are simply called autoregressive (AR(p)), whereas moving-average models (MA(q)) are those with no autoregressive components. The ARMA models of order (p,q) yield superior results to either pure MA or AR forms.

3.2 Artificial neural network

Artificial Intelligence (AI) uses techniques that attempt to follow some elements of the workings of a human brain. Many of such techniques, including hugely popular artificial neural networks (which mathematically can be seen as non-linear regression models), deal with forecasting non-linear time series problems, and they have proven to be an efficient alternative to traditional methods for modeling qualitatively and quantitatively. In this study, we evaluate two different static neural network methods, MLP-NN and RBFNN and one dynamic neural network (IDNN). Hereafter, a brief explanation of all those neural network methods will be introduced.

3.2.1 Static neural network

Multi-layer perceptron

The network architecture of the MLP-NN is shown in Fig. 1. It contains three layers: input, hidden and output layers. Each layer consists of one or more neurons and there are two types of them. First, there are passive neurons that consider the input and output data. Another type is an active neuron that computes data input using Activation Transfer Function (ATF) and produces an output. The most common use of ATF in the hidden and output neuron is the sigmoid function (Zhang et al., 1998; Fernando et al., 2000). The input into an active neuron is a summation of the previous neuron's weighted outputs and the output is a result of applying the sigmoid function to the input. The process is shown in Fig. 1 and the equations for the input and output are

$$\text{input} \quad = \int_{1=0}^{n} \left(W_{ij}, X_{ij}\right) \ \text{ where } \ X_0 = 1 \quad (3)$$

$$\text{output} \quad = \frac{1}{1+e^{-k \ \text{input}}} \quad (4)$$

where x is the output from the previous neuron, w is the weight of the output and k is the slope steepness of the sigmoid function. Extra neuron X_0 is added in the input layer and hidden layer with the output value of 1. This input is called bias and its function is to stabilize computed output between 0 and 1. It does not have a link from the previous neuron.

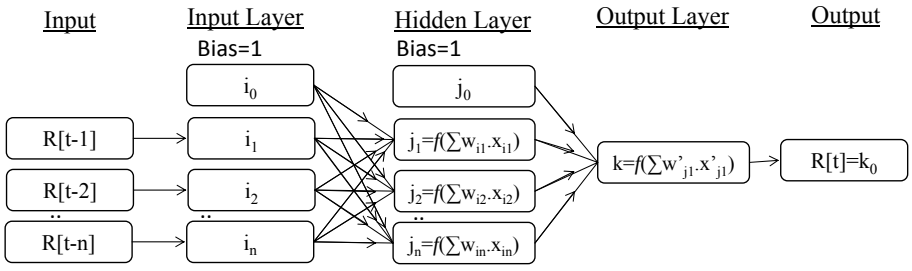


Fig. 1. MLP-NN model architecture.

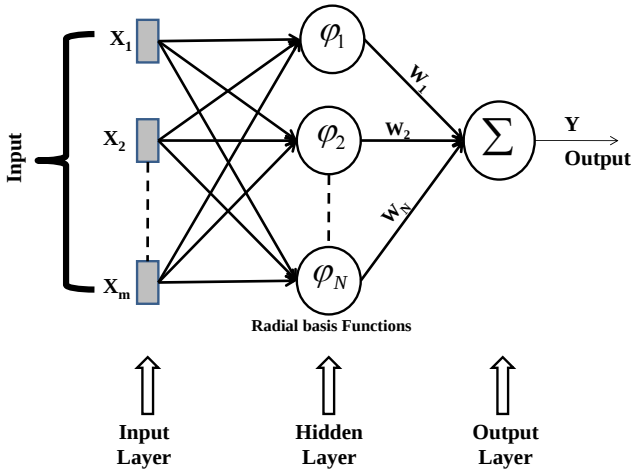


Fig. 2. Architecture of radial basis function neural network.

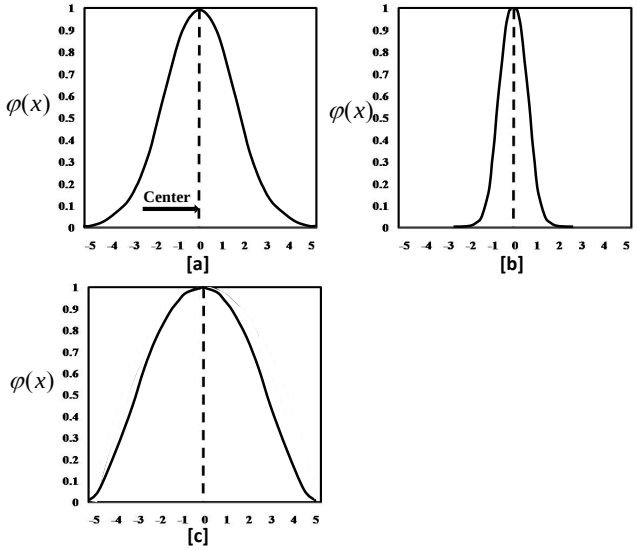


Fig. 3. Radial basis function with different levels of spread. **(a)** normal spread, **(b)** small spread, **(c)** large spread.

Radial basis function neural network

The structure of a RBFNN consists of an input layer, one hidden layer and an output layer (see Fig. 2). The input layer connects the inputs to the network. The hidden layer applies a non-linear transformation from the input space to the hidden space. The output layer applies a linear transformation from the hidden space to the output space (see Orr, 1996).

The radial basis functions φ_1, φ_2, ... φ_N are known as hidden functions, while $\{\varphi_i(x)\}_{i=1}^{N}$ is called the hidden space. The number of basis functions (N) is typically less than the number of data points available for the input data set. Among several radial basis functions, the most commonly used is the Gaussian, which in its one-dimensional representation takes the following form:

$$\varphi(x,\mu) = e^{\frac{\|x-\mu\|^2}{sd^2}} \quad (5)$$

where μ is the center of the Gaussian function (mean value of x) and d is the distance (radius) from the center of $\varphi(x,\mu)$, which gives a measure of the spread of the Gaussian curve.

The hidden units use the radial basis function. If a Gaussian function is used, the output of each hidden unit depends on the distance of the input x from the center μ. During the training procedure, the center μ and the spread d are the parameters to be determined. It can be deduced from the Gaussian radial function that a hidden unit is more sensitive to data points near the center. This sensitivity can be tuned (adjusted) by controlling the spread d. Figure 3 shows an example of a Gaussian radial function. It can be observed that the larger the spread, the less the sensitivity of the radial basis function to the input data. The number of radial basis functions inside the hidden layer depends on the complexity of the mapping to be modeled and not on the size of the data set, which is the case when utilizing multi-layer perceptron ANN (see Ripley,1996; Bishop,1996; and Haykin,1994).

3.2.2 Dynamic neural network

Motivation

The rainfall forecasting models used by KF and/or ARMA assume linear relationships between variables. In addition, the extension of those methods to include stochastic pattern of the rainfall is also linearized in the form of first order difference equations. The non-linear and the non-stationary

parts of the rainfall pattern are not modeled for KF or ARMA and this leads to relatively poor forecasting for the rainfall.

The non-linear AI-based techniques are therefore suggested in this study. Furthermore, the advantage of utilizing the proposed IDNN over the other AI-methods is that the IDNN method performs a temporal processing that gives the model extensive information about the temporal relationship of the input pattern, which is the main challenge in studying the rainfall patterns that incorporate major temporal dimension.

Dynamic networks are generally more adequate than static networks, especially when applied for temporal applications (although they may be somewhat more difficult to train) (Ripley, 1996; Bishop, 1996). Because dynamic networks have memory, they can be trained to learn sequential or time-varying patterns. In order to predict temporal patterns, an ANN requires two distinct components: a memory and an associator. The memory holds the relevant past information, and the associator uses the memory to predict future events. In this case the associator can be a static MLPNN network, and the memory is generated by a time delay unit (or shift register) that constitutes the tapped delay line (Ripley, 1996; Bishop, 1996). Traditional MLPNN is a static and memoryless network that is effective for complex non-linear static mapping, but it or other static neural network type models do not consider explicitly a complete temporal processing since the vector space input encoding gives the model fractional information about the temporal relationship of the inputs (when using current and historical records in the input). In fact, rainfall forecasting is a procedure where previous states of rainfall values have to be explicitly considered. Apparently, rainfall process modeling involves a major temporal dimension and in the ANN context, there are efficient methods to represent and process such models (Haykin, 1994).

Input delay neural network

Figure 4 shows the general architecture of an input delay neural network (IDNN), with the details of the internal structure of a single neuron. The case shown in Fig. 4 considers a tapped delay line that involves the P most recent inputs. In this example, we show three delay elements represented by the operator D. For a case of p, delay elements and an input variable $x(t)$, the network processes $x(t)$, $x(t-1)$, $x(t-2)$, ... and $x(t-p)$, where p is known as the tapped delay line memory length (Haykin, 1994)(in the Fig. 4), and P is equal to 3. Therefore, the input rainfall $R(t)$ to the neuron i (Fig. 4) is given as

$$R_i(t) = \sum_{k=0}^{p} w_i(k)x(t-k) + b_i \tag{6}$$

where $w_i(k)$ is the synaptic weight for neuron i, and b_i is its bias. Then, the output of this neuron (U_i) is obtained by processing $R_i(t)$ by the non-linear activation function $G(.)$, chosen as a sigmoid activation function of neuron i:

$$U_i = G\left(\sum_{k=0}^{p} w_i^{'}(k)x(t-k) + b_i^{'}\right) \tag{7}$$

$$G(R_i(t)) = \frac{1}{1+e^{-R_i(t)}} \tag{8}$$

The output of the IDNN, assuming that it has one output neuron j, a single hidden layer with m hidden neurons, and one input variable as shown in Fig. 4, is given by

$$y_j(t) = F\left(\sum_{i=1}^{m} w_{ji}^{''}U_i + \alpha_j\right) \tag{9}$$

where $F(.)$ is the transfer activation function of the output neuron j (which can be chosen to be a sigmoid or a linear function), αj is its bias and w_{ji} is the weight between the neurons of the hidden layer and the neuron of the output layer.

During the update procedure, we use the Levenberg-Marquardt backpropagation (LMBP). The network training process is performed by providing input-output data to the network, which targets minimizing the error by optimizing the network weights. LMBP uses the second derivative of the error matrix (**E**) to update the weights of the network in a recursive fashion (Haykin, 1994).

The IDNN is a layered feed-forward neural network with appropriate interconnections between the elements in the input layer. This architecture enables it to learn and represent relationships between events in a time sequence. Each sequence provides a pattern to be learned by the network.

Figure 5 shows the architecture of the IDNN. In Figs. 1 and 5 the difference in the input layer for both IDNN and MLP-NN could be observed. The temporal pattern in IDNN is achieved by introducing delays 1 time step through Pt. For simplification, t is dropped and a unit delay operator is represented. The inputs $\boldsymbol{X}_1(t)$,...,$\boldsymbol{X}_N(t)$ represent the N dimensions of the vector $\boldsymbol{X}_b(t)$ (in our application the past rainfall records) at time t and P_1 is the number of the successive vectors $\boldsymbol{X}_b$, which defines the temporal window size to which the neuron reacts. These N inputs are multiplied by several weights ($W_j(0)$,...,$W_j(P)$, j_1,...,N), one for each delay and one for the undelayed input and the weighted sum, Y, is passed through the sigmoid function. For P_3, and N_3, for example, 12 weights are needed to compute the weighted sum of these inputs, with each input measured at four different points in time. In this way, the IDNN unit has the ability to relate and compare current input to the past history of events.

Figure 5 shows an example of IDNN architecture. The input layer is constituted of three successive input vectors having $\boldsymbol{X}\_3$ dimensions. The input layer is fully interconnected to the first hidden layer of four time delay hidden units, where

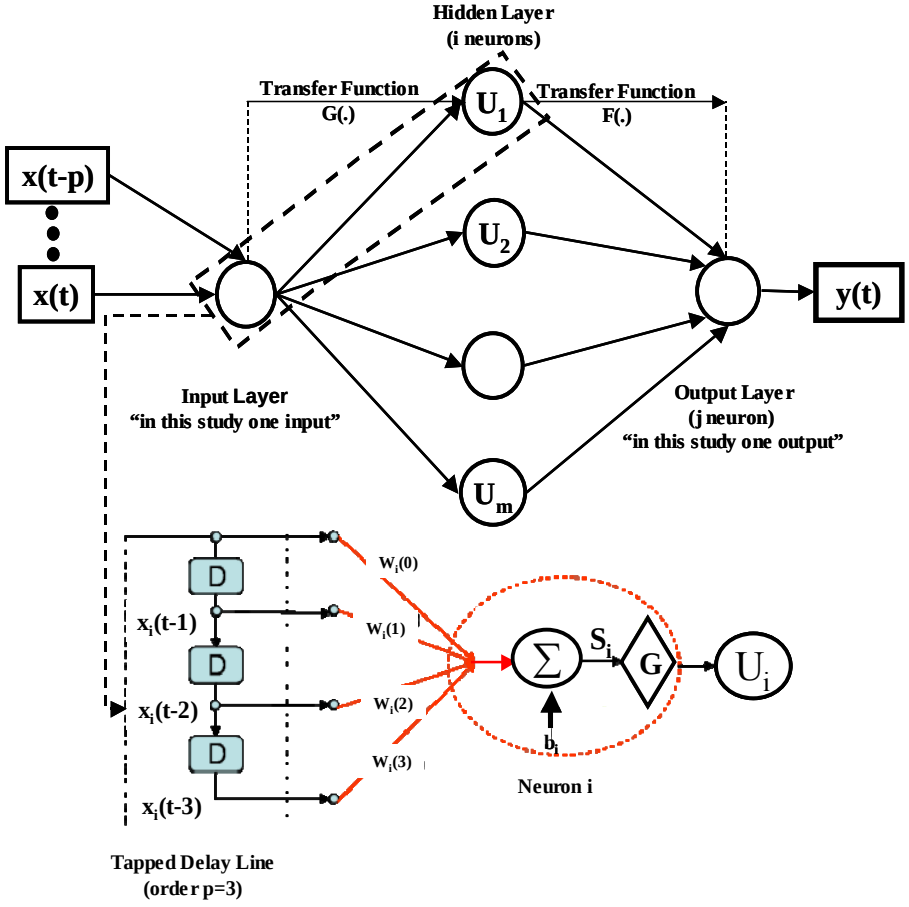


Fig. 4. Input delay neural network architecture and single neuron calculations.

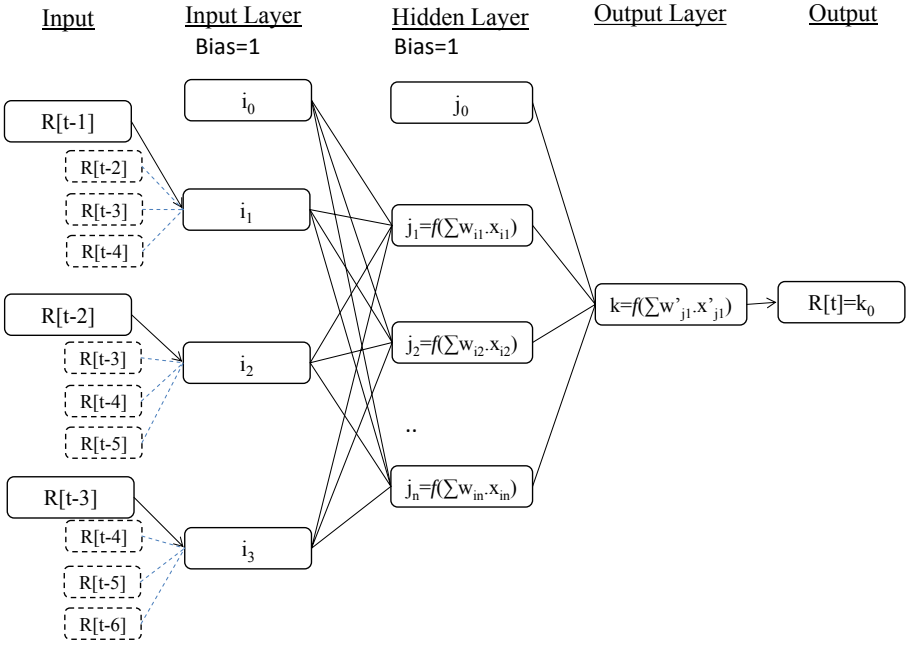


Fig. 5. IDNN model architecture.

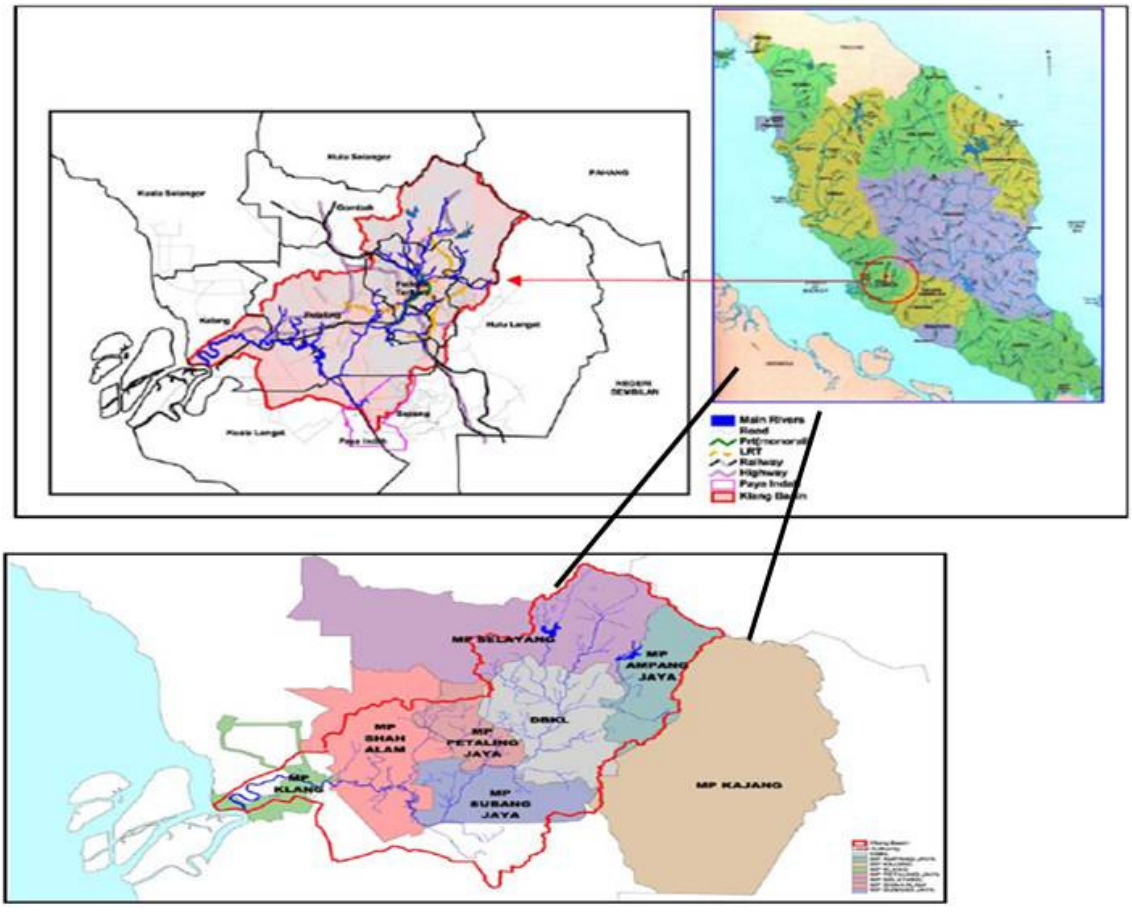


Fig. 6. Local authorities within Klang River Basin.

$\boldsymbol{X}\_j\_3$ and $\boldsymbol{P}\_3$ (i.e. three inputs over four steps with time delay 0, 1, 2 and 3). The temporal window, whose size $\boldsymbol{P}\_1\_3$ is shown as the shaded area, steps down the time axis and $Wij(0)$ to $Wij(2)$, j_1,...,3 represent the weights of unit i.

3.2.3 Study area and data collection

The Klang River Basin is located on the west coast of Peninsular Malaysia and encompasses the Federal Territory of Kuala Lumpur, parts of Gombak, Hulu Langat, Klang, and Petaling districts in Selangore Stats, and the municipal areas of Ampang Jaya, Petaling Jaya, and Shah Alam. Klang is geographically located at latitude (3.233 degrees) 3°13′58″ North of the equator and longitude (101.75 degrees) 101°45′0″ East of the Prime Meridian on the map of Kuala Lumpur. The study area location map is shown in Fig. 6.

The Klang River originates in the mountainous area about 25 kilometers (km) northeast of Kuala Lumpur. It is joined by 11 major tributaries while passing through the Federal Territory and the area downstream of Kuala Lumpur, before joining the Straits of Malacca at Port Klang. The Klang River has a total length of about 120 km. The basin is 1290 square kilometers, about 35 percent of which has been developed for residential, commercial, industrial, and institutional use. The upper catchments of the Klang River and its tributaries – the Gombak and Batu Rivers – are covered with well-maintained forests. However, the lower reaches of the basin, with extensive urban land development activities, are major contributors of sediment load and flood peaks (Tan, 2009; Hiew, 1996; Gibson and Dodge, 1983).

It is also characterized by uniform high temperature, high relative humidity, heavy rainfall and little wind. All information and data that are available about Klang River were based on Klang Gates Dam data. For this study, the data used were from years 1986 to 2008 (monthly basis) and between 1997 and 2008 (weekly basis). The available data for catchment are divided into two groups: training set (calibration) and a testing set (validation). The rainfall data statistics have been investigated, including maximum, minimum and mean averages. The average annual rainfall depth in the study area is about 2400 mm. The highest rainfall occurs in the month of April and November with a mean of 280 mm. The lowest rainfall occurs in the month of June with a mean of 115 mm. The rainfall data on a monthly and weekly basis are shown

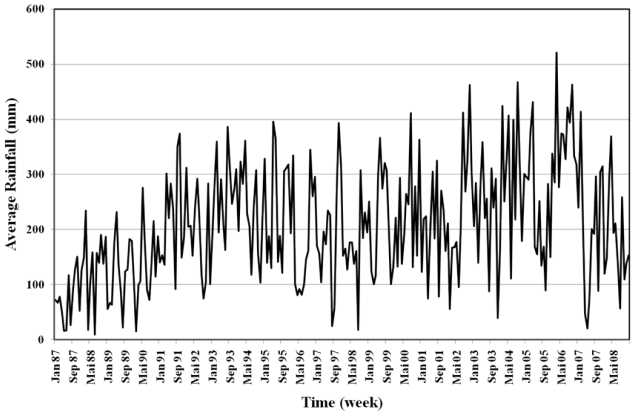


Fig. 7. Monthly actual rainfall records on Klang River on for period 1987–2008.

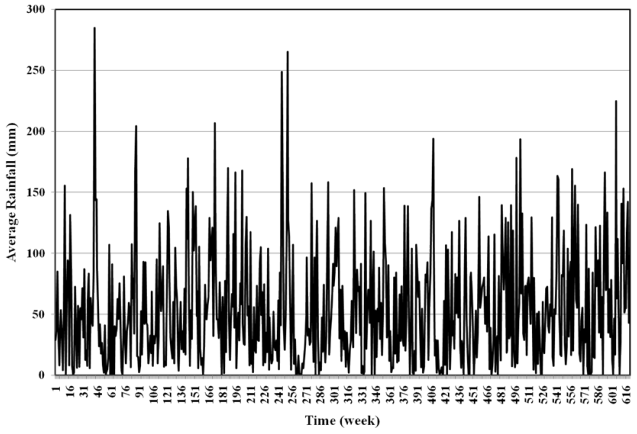


Fig. 8. Weekly actual rainfall records on Klang River on for period 1997–2008.

in Figs. 7 and 8, respectively (Tan, 2009; Hiew, 1996; Gibson and Dodge, 1983). For monthly rainfall forecasting, total monthly rainfall data is 261, containing 237 samples used for training (number of training records is 237 –$(P+1)$) and another 24 samples used to test the generalization ability of the networks. However, for weekly forecasting, total weekly rainfall data is 621, containing 571 samples used for training (number of training records is 571 –$(P+1)$) and the rest containing 50 samples used to test the generalization ability of the networks.

One of the steps of data preprocessing is data normalization. The need to make harmony and balance between network data range and an activation function used causes the data to be normal in activation function range. Sigmoid activation function is used for all layers. By considering Sigmoid, it can be seen that the range is between 0 and 1, so data must be normalized between 0 and 1. (Eq. 8) The following formula was used:

$$x_n = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{10}$$

where x is actual rainfall data and $x_{\min}$ is minimum value of original series and $x_{\max}$ is maximum value of original series.

4 Methodology

Most neural network approaches to the problem of forecasting use a multi-layer network trained using the backpropagation algorithm. Consider a time series $x(1), \ldots x(t)$, where it is required to forecast the value of $x(t+1)$. The inputs to the multi-layer network are typically chosen as the previous w (w defined here as input window) values $x(t-w+1), \ldots, x(t)$ and the output will be the forecast. The proposed network model is accomplished on sufficient training and testing data sets that are extracted from the historical time series. In addition to previous time series values, one can utilize as inputs the values or forecasts of other time series (or external variables) that have a correlated or causal relationship with the series to be forecasted. For our rainfall forecasting problem such time series could be the temperature and relative humidity at the river basin. For the majority of forecasting problems, such external inputs are not available or are difficult to obtain. As is the case with many neural-network applications, preprocessing the inputs and the outputs can improve the results significantly. Input and output preprocessing means extracting features from the inputs and transforming the target outputs in a way that makes it easier for the network to extract useful information from the inputs and associate it with the required outputs. Preprocessing is considered an "art" and there are no set rules to choose it. Even some very intuitively appropriate transformations may turn out no value when checking the actual results. For our case the main inputs are the previous time series values. We have used normalization as a preprocessing of the inputs (as described Sect. 2.3).

4.1 Model structure

Generally, formation of an appropriate architecture of a neural network for a particular application is an essential step and issue since the network topology directly affects not only its computational complexity and its generalization capability, but also the accuracy level. Different theoretical and experimental studies have shown that larger-than-necessary networks tend to over-fit the training samples and thus have poor generalization performance and a low accuracy level for the unseen data, while too-small networks (that is, with very few hidden neurons) will have difficulty learning the training data. Currently, there is no established methodology for selecting the appropriate network architecture prior to training. Therefore, we resort to the trial-and-error method commonly used for network design. In addition, the performance goal (mean square error MSE) for the model during

the training stage was forced to be 10^{-4}; thus, the neural network is guaranteed to hedge over-fitting the training data.

One more important step in the model implementation, especially in the multi-variate ANN forecasting context, is the selection of appropriate input variables, since it provides the basic information about the system considered. Bowden et al. (2005) developed a hybrid genetic algorithm and general regression neural network (GAGRNN) in order to determine which inputs have a significant relationship with the output (dependent) variable. Such a method is suitable when comprehensive data for different hydrological and metrological parameters are available for the study area. In the current study, the available data are from the historical rainfall records; thus, different input pattern in terms of the length of the previous rainfall records (window size = WS) have been examined. Five window sizes (WS = 1, 2, 3, 4 and 5) were considered in this study. In fact, other hydrological and metrological parameters that affect the rainfall intensity are not available at the study area, thus, the proposed rainfall forecasting model structure is designed based on the historical rainfall model. Searching for the best WS is evaluated via two statistical indexes during training to determine the relative importance of each WS on the model accuracy level and generalization. Details about the model performance evaluation will be described in the following section.

For the monthly basis rainfall forecasting model, two different scenarios(A and B) are examined. The first scenario A is developed using the following structure:

$$R(t,z) = f\,(R(t,z-1), R(t-1,z), \ldots R(t-w,z)) \qquad (11)$$

where $R(t,z)$ is the rainfall in a certain month (t) and year (z),

while the second scenario B is structured based on the following structure:

$$R(t,z) = f\,(R(t-1,z), R(t-2,z), \ldots R(t-w,z).) \qquad (12)$$

For the weekly basis rainfall forecasting model, the model is configured as scenario B for the monthly-based model.

A common method for presenting the input data to the model is to consider a sliding (or moving) window of input sequences. This approach has been widely used with the standard MLP-NN. In this case, a fixed number of past items of information is selected and introduced to the input layer of the network. For instance, if it is required to model the rainfall pattern based on the input at the present instant in time and the past two samples (window size = 3); the MLP-NN input layer should have three input neurons (see Fig. 9). Therefore, the network is provided with a static memory, which is considered as a limitation of the MLP-NN.

The time line index for the proposed model process is presented in Fig. 10. The upper part of Fig. 10 shows the process while utilizing the static neural network (MLP-NN and RBFNN) models.

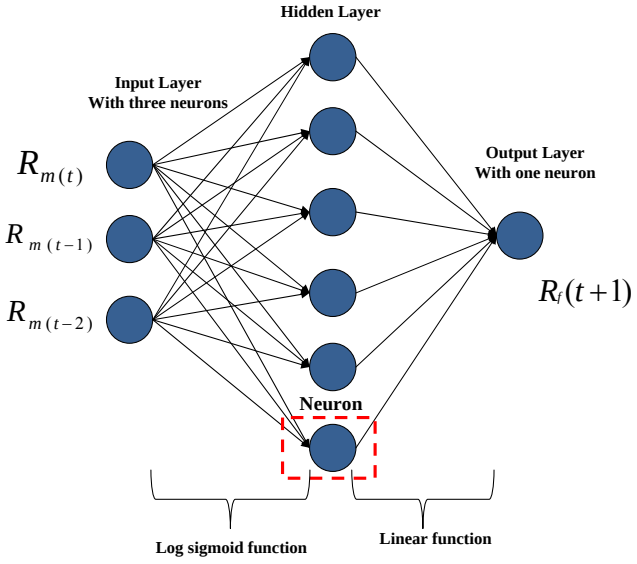


Fig. 9. Neural network model architecture utilized for rainfall forecasting.

The IDNN model with a sliding window input sequence is shown in the lower part of Fig. 10. In this study, one and two time step input delay sequences will be considered. The second-order delay effect will be considered by training the IDNN model to experience, in the input layer, the previous one time step sample, in addition to the present rainfall record. Moreover, the higher-order error can be considered by having two and three time step delay inputs. In Sect. 4.3, the impact of using one and two input delay elements will be demonstrated and discussed.

It should be noted here that a static neural network with a four input pattern (window size = 4) is not similar to a dynamic neural network with a three input pattern (window size = 3) with a one-time step input delay. This is due to that the dynamic neural network incorporates the associator (network weights and bias) procedures at the current time step and memorizes the past information (network weights and bias) from the previous time step, while the static neural network only handles certain time steps with longer input patterns.

Recall that our collected rainfall data spanned the period from 1986 to 2008 on a monthly basis and from 1997 to 2008 on a weekly basis, so the forecasting is performed for these two time horizons.

4.2 Training algorithm

The neurons in the network architecture are interconnected between the layers. The computation starts from input neurons where data input is received and then propagates to hidden neurons and further to output neurons. A neuron in the output layer produces model output. If the number of neurons and layers are established, the only unknown parameter

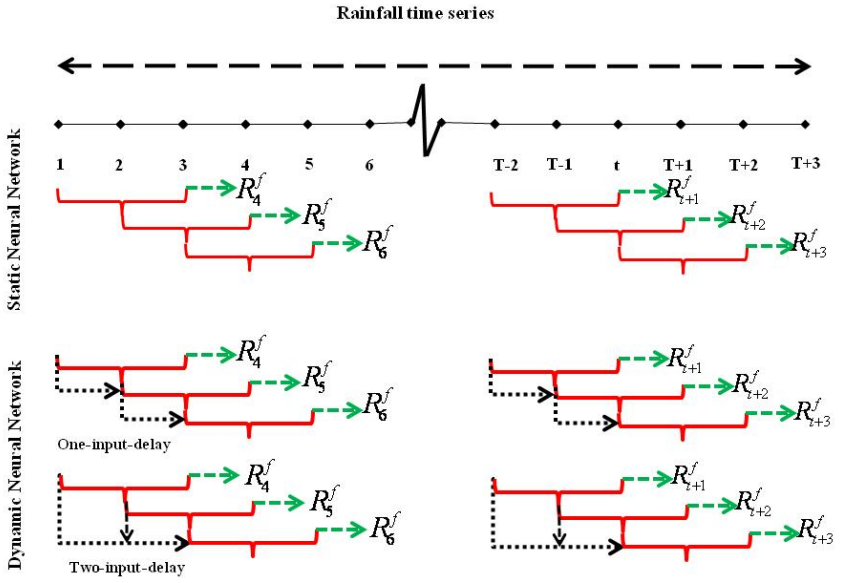


Fig. 10. Model time line index with sliding window method.

in the computation is the weights. The process of data training determines the weights. Data training is a process of feeding sample historical data to the input and output of the network model so that the network model can simulate the sample data. The data training process involves feed-forward and backpropagation computation cycles. The backpropagation computation is an adjustment of output and hidden neuron weights based on the gradient descent method. These weights are normally initialized with random values to speed up the data training process to a solution.

For optimization purposes, we use a backpropagation variation – namely the Levenberg-Marquardt backpropagation (LMBP) – for the IDNN training. This method uses the approximate Hessian matrix in the weight update procedure as follows:

$$\Delta W_{\mu} = -[\mathbf{H} + \mu I]^{-1}\mathbf{J}^{t} r \tag{13}$$

where r = residual error, μ = variable small scalar that controls the learning process, $\mathbf{J} = \Delta E$ = Jacobian matrix, E = cost function, and $\mathbf{H} = \mathbf{J}^{t}\,\mathbf{J}$ denotes the approximate Hessian matrix. In practice, this method has been found effective in finding better optima than standard backpropagation and the conjugate gradient descent method (Hagan and Menhaj, 1994). A detailed description of this algorithm is given by Masters (1995). Here, the LMBP algorithm is also used to train the all proposed network models.

Upon successful data training, data forecasting can be made to new data input. To evaluate forecasting performance, validation data are fed only to the input of the network model where single feed-forward computation computes the data. The output of the computation is the model output. Several performance measures are applied to model outputs and observed outputs from the validation data set to determine the accuracy and reliability of the network model developed.

4.3 Model performance criteria

To compare and evaluate the effectiveness of rainfall forecasting model applied at Klang Gates Dam, models were assessed on the basis of important performance measures. Although, all models achieved a MSE of less than 10^{-4} during the training process, it is important to examine the model performance utilizing different input sequences and patterns. Consequently, statistical analysis for the model output in the testing session utilizing the inflow data for the period between 1998 and 2003 was carried out in order to evaluate the model performances. To analyze the fittingness of forecasted inflow with the natural inflow during the testing period, two statistical measures were used to examine the goodness of fit of the proposed models methods to the testing data. These measures include the RMSE (Root Mean Square Error) and the maximum relative error (RE) to examine the relative accuracy of both models for each inflow event as represented by Eqs. (7) and (8).

$$\mathrm{RMSE} = \left(\frac{1}{N}\sum_{1}^{N}(R_{\mathrm{f}} - R_{\mathrm{m}})^{2}\right)^{0.5} \tag{14}$$

$$\mathrm{MaxRE} = \max\sum_{n=1}^{n}\left(\frac{R_{\mathrm{m}} - R_{\mathrm{f}}}{R_{\mathrm{m}}}\right) * 100 \tag{15}$$

where R_{f} is the rainfall at Klang River, R_{m} is the actual rainfall and N is the number of the months/weeks.

In fact, in developing such a forecasting model using a neural network, the model could perform well during the

training period and might provide higher levels of error when evaluating during either a validation or a testing period. In this context, in this study the authors used these performance indices to make sure that the proposed model could provide a consistent level of accuracy during all periods. The advantage of utilizing these two statistical indices as performance indicators of the proposed model is first, to make sure that the highest error while evaluating the performance is within the acceptable error for such a forecasting model. Utilizing the RMSE is to ensure that the generalized error distribution within the validation period is not high. Consequently, examining these two indices together as model performance indicators during training is to guarantee that the model could relatively provide the same level of accuracy while examining the model for unseen data in the testing stage.

In addition, as the forecasting accuracy of the peak and low inflow events is of particular interest of the reservoir operation, it is important to evaluate the model performance considering these inflow events. In order to assess the model performance during these events, another two error criteria are also utilized: the peak flow criteria (PFC) and low flow criteria (LFC), which can be computed by Eqs. (12) and (13).

$$\mathrm{PFC} = \frac{\left(\sum_{i=1}^{T_\mathrm{p}} \left(R_\mathrm{m} - R_\mathrm{f}\right)^2 \cdot (R_\mathrm{m})^2\right)^{0.25}}{\left(\sum_{i=1}^{T_\mathrm{p}} (R_\mathrm{m})^2\right)^{0.5}} \tag{16}$$

$$\mathrm{LFC} = \frac{\left(\sum_{i=1}^{T_i} \left(R_\mathrm{m} - R_\mathrm{f}\right)^2 \cdot (R_\mathrm{m})^2\right)^{0.25}}{\left(\sum_{i=1}^{T_i} (R_\mathrm{m})^2\right)^{0.5}} \tag{17}$$

where T_p = number of peak rainfall greater than one-third of the mean peak rainfall observed; T_i = number of low rainfall lower than one-third of the mean low rainfall observed. Coulibaly (2001) reported that both PFC and LFC provide better performance indicators for assessment of the forecasting model performance for the extreme rainfall events. As the model can provide low PFC or LFC, the model represents better fit.

One more index is examined for the proposed model, which is evaluating the consecutiveness of the rainfall. In fact, the model could provide a relatively good fit in terms of the error values; however, the forecasting value might not follow the sequence changes of the rainfall pattern values (whether the rainfall increased or decreased), for example, in case the difference between two consecutive actual values of rainfall is positive, $[R_\mathrm{m}(t+1) - R_\mathrm{m}(t) > 0]$. On the other hand, if examined using the forecasted value, it could be negative $[R_\mathrm{m}(t+1) - R_\mathrm{m}(t) < 0]$, where $R_\mathrm{f}(t+1)$ is the forecasted value by the proposed rainfall forecasting model. In fact, the error value could be same value but with a different sign, $\pm$. Even the model might provide a forecasted value with a relatively small error value; it might be with a different sign, which is considered as a major drawback of the model performance and shows mismatching with the real rainfall pattern series.

In some cases, the value of RE (Eq. 15) could be relatively small and within the accepted level of accuracy and that shows that the model performs well. However, the R_f value could show the rainfall will increase, but in reality the rainfall will decrease or vice versa.

All the development made on this study was implemented using MATLAB computer-aided design software (Beale and Demuth, 2001). The neural network toolbox of MATLAB was utilized and the code was set up to include all the above procedures.

5 Results and discussions

The forecasting model architecture described in Sect. 4 was applied on monthly and weekly rainfall data at Klang River, Malaysia. In fact, the procedure of the study began with utilizing the MLP-NN method searching for best model configuration in terms of input pattern (window size), and then used this window size in the other methods. In this way we make sure that the comparative analysis between all proposed methods was adequately performed.

All networks successfully achieved the target MSE of 10^{-4}. For example, the training curve utilizing the MLP-NN method for the weekly data is demonstrated in Fig. 11, showing convergence with the target MSE after 563 iterations. This section is organized to present the results for each method individually, followed by an evaluation of the optimal model based on the model performance measures presented in Sect. 4.3.

5.1 Forecasting utilizing MLP-NN

Several trials and error in order to search for the optimal MLP-NN architecture have been carried out. One and two hidden layers and number of neurons, ranging between one to ten, different transfer function (tan sigmoid, log sigmoid, linear) and finally different window sizes ($w = 1$ to $w = 5$) have been examined in order to attain the optimal model configuration. In fact, the procedure was performed by implementing all trials for number of hidden layers, number of neurons in each layer and the type of transfer function, while keeping the window size unchangeable. Then, we repeated all the trials again in the other window size. Such a procedure was applied for both weekly and monthly time horizon as the objective for this study.

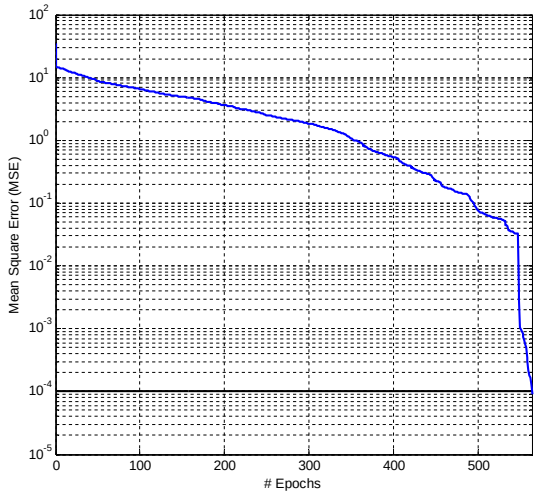


Fig. 11. Training curve for monthly basis using the MLP-NN model.

In this context, the model configuration that provides the best performance, in terms of lower maximum relative error and RMSE while in the training procedure, is selected. For the weekly basis horizon, the optimal model configuration is achieved when the window size = 4 (number of neurons in the input layer), two hidden layers with number of neuron equal to 8 and 5, respectively, and log sigmoid transfer function occurs between input layer to hidden layer #1 and from hidden layer #1, #2 and linear transfer function between hidden layer #2 and the output layer. On the other hand, the optimal architecture for the monthly basis horizon is attained when window size = 3, with one hidden layer with 7 neurons. The transfer functions are tan sigmoid and linear between input layer and hidden layer #1 and from hidden layer #1 to output layer, respectively.

In order to show how the trial and error procedure for selecting the best parameter set of certain ANN architecture was performed, an example for weekly basis is presented in Fig. 11. For better visualization, the inverse value of both RMSE and maximum error were used (as seen in Fig. 11b and c) instead of the real values, (Fig. 11a shows the real value for both indices). Figure 12 shows the changes in the value of the RMSE and the maximum error versus the number of neurons when the number of hidden layers is one, as shown in Fig. 12a and for two hidden layers, as shown in Fig. 12b (RMSE) and Fig. 12c for the maximum error during the training. It is interesting to observe the large number of local minima that exist in both domains. It can be observed that the best combination of the proposed statistical indices for evaluating forecasting model for the weekly basis is when the ANN architecture has 8 neurons in the first layer and 5 neurons in the second layer, achieving RMSE 37.2 mm and maximum error 50 %.

With the purpose of examining the performance for both scenarios A and B for the monthly basis rainfall forecasting

Table 1. Maximum RE% and correlation coefficient associated with the output of MLP-NN model on a weekly/monthly basis for years 2007 and 2008.

Model	Max RE%		Correlation Coefficient	
	2007	2008	2007	2008
Scenario A	84	87	0.51	0.47
Scenario B	70	65	0.64	0.68

model-based MLP-NN method, Table 1 illustrates the performance measures for both scenarios. It can be observed that scenario B outperformed scenario A and provides a better accuracy level. It might be due to that scenario A includes the rainfall record of the same month of the previous year, which might not correlate and relate to rainfall values of the same month in the current year, and thus, the model scenario A provides relatively poor forecasting accuracy.

Figure 13 shows the performance of monthly and weekly rainfall forecasting using the MLP-NN model. Figure 13a shows the RE for the monthly basis forecasting data used for training; it could be depicted that the maximum RE is 25 % while the RMSE is 55.6 mm. However, the performance for the unseen data during the testing stage is about 65 %, as the maximum RE and RMSE equal to 79.89 mm(see Fig. 13b). It should be noticed here that RE during the testing is almost 3 times that of one experienced during the training stage.

On the other hand, Fig. 13c shows the MLP-NN model while examining the data on a weekly basis during the training. It could be observed that, although the model provides maximum RE at 50 % during training (which is relatively high if compared with the case for monthly basis), the performance of the model during the testing stage (as shown in Fig. 13d) is also within the same range (except one odd case at week #17, RE equal to 80 %), which is not the case for the model on a monthly basis. Such an observation shows that the model for weekly basis provides higher consistency level over the monthly basis; it might be due the fact that the model for weekly basis incorporates large data records for training that allow the model to mimic several patterns and able it to provide the same level of accuracy during testing. Such observations could be confirmed when examining the RMSE during the training and testing stages, which are 37.2 mm and 43.5 mm, respectively.

Moreover, to compare the ANN to existing modeling techniques, we compared the prediction error of the ANN with the prediction error from the autoregressive-moving-average (ARMA) models. Table 2 shows the comparison of the performance of the ANN to the ARMA models, using the maximum relative error (MaxRE%) (as shown in Eq. (11)) and the correlation coefficient (Elshafie and Noureldin, 2011) indicator for each month. It is obvious from Table 2 that the ANN model outperformed the ARMA models with

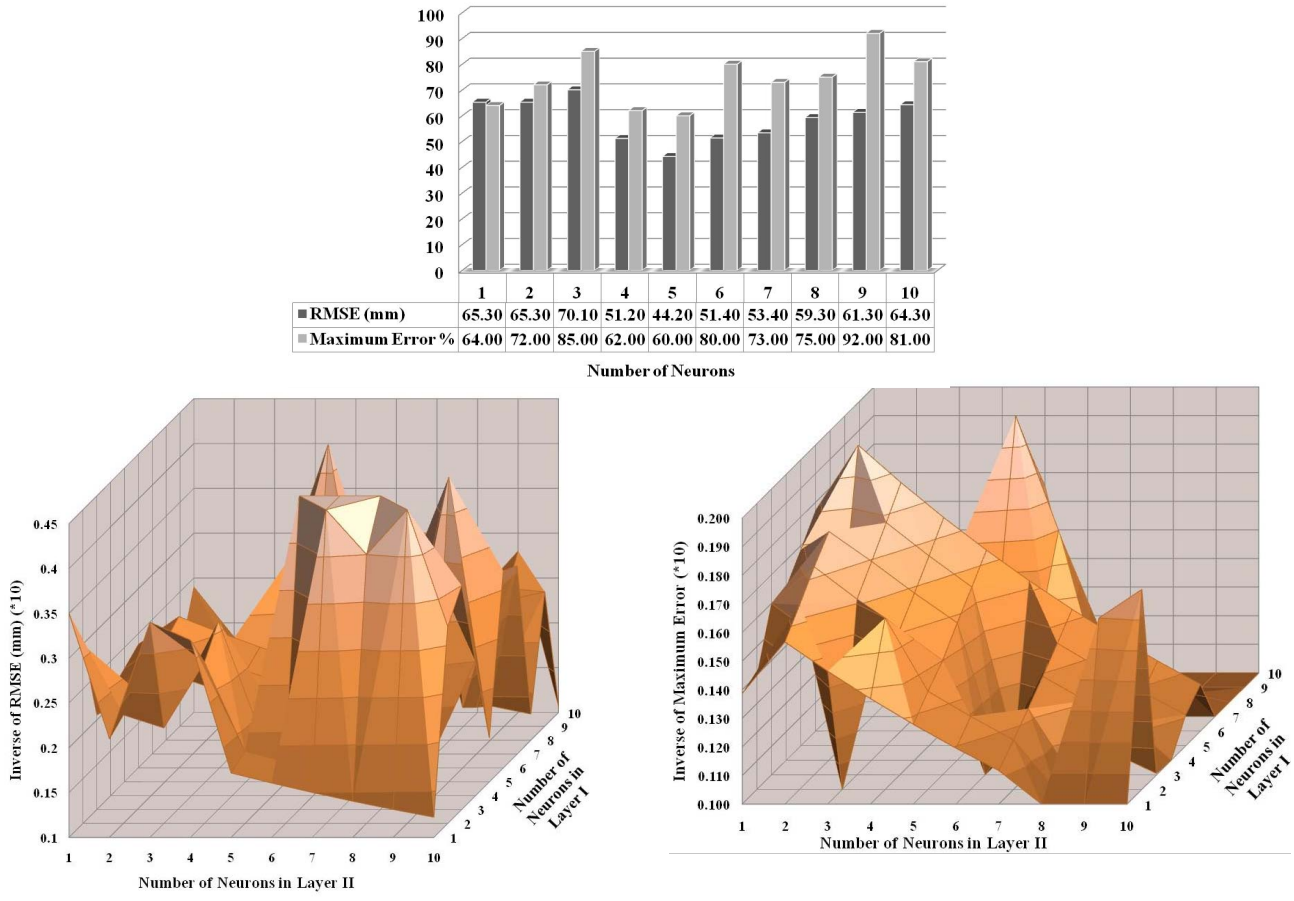


Fig. 12. Neural network performance "RMSE and maximum error%" utilizing different architecture, **(a)** one hidden layer **(b)** and **(c)** two hidden layers.

Table 2. Maximum RE% and correlation coefficient associated with the output of ANN and ARMA models on a weekly/monthly basis for years 2007 and 2008.

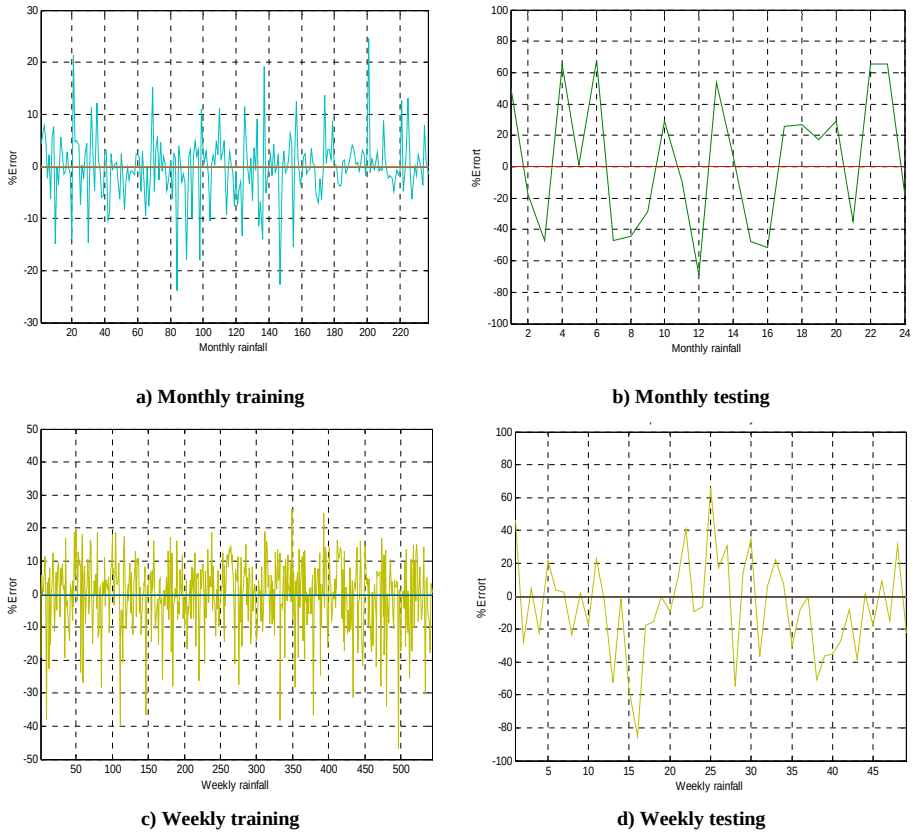
Model		Max RE%		Correlation Coefficient	
		2007	2008	2007	2008
ARMA Model	Weekly	62	84	0.63	0.54
	Monthly	81	74	0.54	0.61
ANN Model	Weekly	50	78	0.71	0.58
	Monthly	70	65	0.64	0.68

remarkable improvements in the correlation coefficient at both the weekly and monthly basis.

5.2 Forecasting utilizing RBFNN model

Keep in mind that the optimal window size achieved, based on MLP-NN model, 3 and 4 for monthly and weekly, respectively. It should be noticed here that the architecture of RBFNN network is quite simple if compared with MLP-NN. Once the window size (input pattern) was resolved, adjustment of the spread of the RBFNN model configuration is (as described in Sect. 2.1.2) the only parameter to be obtained. The spread (step size) is achieved by trial and error as well. The optimal values of the spread were found to be equal to 0.07 for the monthly and 0.03 for the weekly model.

Figure 14 illustrates the accomplished results for the monthly and weekly rainfall forecasting using the RBFNN model. For monthly basis, as demonstrated in Fig. 14a, the RE during training is slightly increased if compared with



a) Monthly training **b) Monthly testing**

c) Weekly training **d) Weekly testing**

Fig. 13. Performance of the MLP-NN model for monthly and weekly rainfall forecasting during training and testing stages.

MLP-NN; however, the RE level is improved for the testing data (Fig. 14b). It could be observed that the maximum RE is within ± 40 %, which means considerable improvement over the MLP-NN model. In addition, the RMSE is slightly improved to be 68.7 mm, which is almost 90 % of similar value when using MLP-NN.

For weekly basis horizon model, Fig. 14c shows the performance of the RBFNN during training. If Fig. 14c is carefully examined, it could be observed that the pattern of RE is similar to RE pattern using MLP-NN model, but the RE value is relatively improved. Consequently, the performance for the RBFNN model, in terms of RE, is also enhanced when examining the testing data and if compared with MLP-NN model (see Fig. 14d).

5.3 Performances of IDNN model

Similar procedures were applied for the rainfall forecasting utilizing the IDNN model. To investigate the effect of temporal dimension value of the rainfall($t+1$) (the output of the IDNN module) on the present and past rainfall pattern inputs (the input to the IDNN module), we examined the performance of the IDNN model, using one time input delay element to the case of two input delay elements. In the case of one time input delay, Fig. 15 shows the performance for monthly and weekly basis horizon. As can be depicted from Fig. 15a, significant enhancements took place in the forecasting accuracy in terms of the relative error. The maximum RE does not go behind 12 %, which is almost one-third of the maximum RE experienced using MLP-NN and RBFNN. In addition, it is noticeable that a considerable improvement in the maximum RE (± 20 %) for testing data is achieved, as shown in Fig. 15b. Similar enhancement while applying the IDNN for the weekly basis data could be observed as shown in Fig. 15c and d.

On the other hand, in the case of using two-time input delay, Table 3 shows the maximum RE and RMSE for both training and testing stages for monthly and weekly basis utilizing one and two input IDNN architecture. The results clearly show that utilizing two-time input delay elements has insignificant improvements to the model performance if compared to the one-time input delay IDNN architecture, especially for the weekly basis model. While the proposed IDNN-based model showed a slight accuracy improvement when using two-time input delay elements instead of one for the monthly basis model, the additional delay element significantly complicated the training procedure.

In light of the results presented above, apparently the highest RMSE is of the MLP-NN model in monthly rainfall forecasting and it is equal to 79.89 mm, whereas the smallest

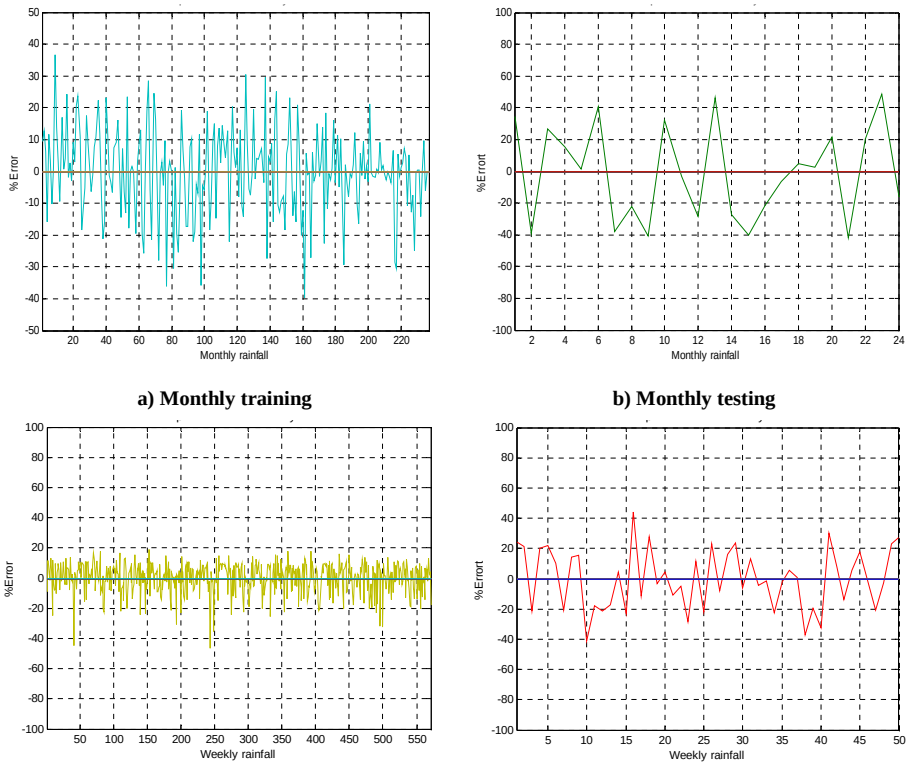


Fig. 14. Performance of the RBFNN model for monthly and weekly rainfall forecasting during training and testing stages.

Table 3. RMSE and maximum RE for monthly and weekly rainfall forecasting model utilizing IDNN.

Type of model	RMSE (mm)				Maximum RE%			
	Monthly		Weekly		Monthly		Weekly	
	Train	Test	Train	Test	Train	Test	Train	Test
IDNN One-time step	9.2	30.3	2.2	7.3	7.4	20.89	7.1	17.1
IDNN two-time step	9.1	28.2	2.4	7.2	7.2	19.1	8.1	19.3

value of RMSE is of the IDNN model in the weekly rainfall forecasting model with only 7.3 mm. It could be remarked that, generally, the performance for the weekly rainfall forecasting is better than monthly rainfall forecasting. This is due to the inadequacy of historical data records on the monthly basis, which is 261 records, while for weekly, historical 621 rainfall records; thus, the model could capture most of temporal dimension of rainfall pattern and be able to provide lower forecasting error. In addition, it is obvious that the optimal results were received when using the IDNN method. Furthermore, with one-time step input delay, the IDNN is sufficient to achieve a significant level of accuracy. With respect to this observation, in Klang River Basin rainfall pattern, it might be the temporal dimension feature of rainfall that is in second order level. However, it could be inadequate for other river basins that might require introducing a higher order level.

For further assessment, the IDNN model with one-time step input delay was examined for the peak and low rainfall events, so that the comparisons between the forecasted and actual rainfall values are visually presented using the 45° line and two deviation lines, with ±15 % deviation from the 45° line and they demonstrate the low, average and peak rainfall ranges for both monthly and weekly basis as shown in Figs. 16 and 17, respectively. The scatter plot of forecasted rainfall-based monthly (as depicted in Fig. 16) is a little distant from the ideal line in the case of the high rainfall, while it is closer to the ideal line in the low rainfall range. The same rainfall forecasting features for weekly basis could be observed from Fig. 17. Apparently, the model provides better accuracy for low rainfall for either low or

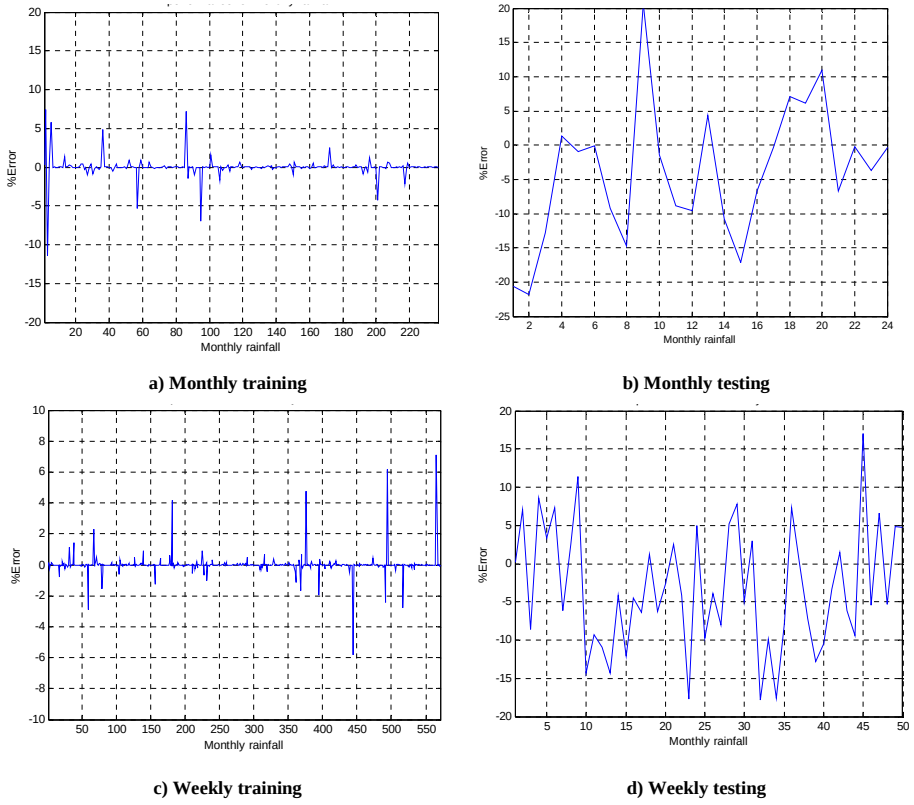


Fig. 15. Performance of the IDNN model for monthly and weekly rainfall forecasting during training and testing stages.

high rainfall seasons. This is due to the fact that the peak rainfall events for both low and high rainfall seasons were not experienced adequately during training period. In order to validate the previous analysis of the model performance in providing an accurate inflow forecasting for the peak and low inflow events, the PFC and LFC statistics, as discussed above in Sect. 4.3, are presented in Table 4. As presented in Table 4, it can be observed that the developed model can perform the function of providing an accurate rainfall forecasting at Klang River for even the extreme rainfall events, with error that does not go above 9.2 % of the actual rainfall.

Finally, the IDNN model with one-time step in the input delay was evaluated for its ability to model the consecutive rainfall pattern(see Sect. 4.3). Table 5 shows the results for this evaluation index. The negative values mean that the model failed in matching the rainfall consequences and visa versa for the positive value. It could be observed that the IDNN model for weekly basis forecasting has outperformed the other methods.

In light of the above discussion, it could be mentioned that the use of the input delay lines improves the standard MLP forecast accuracy. This means that the IDNN model has taken advantage of the input data at different time steps to improve the forecast accuracy. However, the IDNN remains less accurate for the peak and low flows forecasting. This may indicate the limitation of the IDNN network since there is no possible modification of the fixed time delays. Therefore, the IDNN may not be appropriate where accurate peak and low rainfalls forecastings are needed. It may also point to the fact that IDNN architecture with fixed time delays may be not very well-suited for modeling a time-varying process such as the hydrology extreme events.

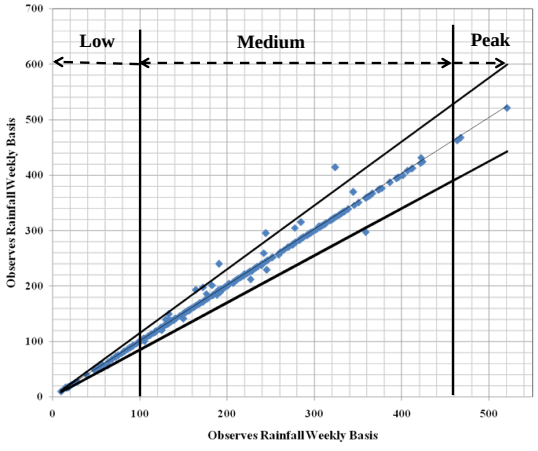


Fig. 16. Forecasted and actual rainfall at Klang River (weekly basis).

Table 4. IDNN performance based on the peak and low flow error criteria for monthly and weekly forecasting.

Year		PFC (%)	LFC (%)	Average (%)
Monthly	Train	4.20	1.80	3.00
	Test	6.15	3.07	4.61
weekly	Train	8.50	2.40	5.45
	Test	9.20	6.30	7.75

Table 5. Models performance for matching the rainfall consequences.

Stage	Method	Monthly		Percentage	Weekly		Percentage
		Positive	Negative	of corrected (%)	Positive	Negative	of corrected (%)
Train	MLP-NN	240	20	92	522	48	92
	RBFNN	248	12	95	528	42	93
	IDNN	235	7	90	553	17	97
Test	MLP-NN	15	9	63	38	12	76
	RBFNN	17	7	71	41	8	82
	IDNN	20	4	83	49	1	98

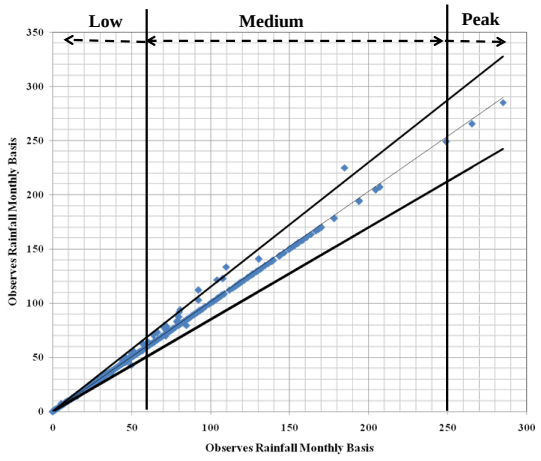


Fig. 17. Forecasted and actual rainfall at Klang River (monthly basis).

6 Conclusions

This study is focused on modeling the temporal dimension of the rainfall pattern in order to achieve better rainfall forecasting results. In this context, this study investigated three different neural networks. The proposed models were implemented for offering a rainfall forecasting model on Klang River Basin for monthly and weekly time horizon. The results reveal that the dynamic (temporal) neural network, namely IDNN, could be suitable for modeling the temporal dimension of the rainfall pattern, and thus, provide better forecasting accuracy. Our results show that IDNN model, with one-time step input delay for weekly basis rainfall forecasting, achieved the highest accuracy level. This technique could also be applicable to other studies in other river basins with different time step input delay, according to how far is the temporal dimension of rainfall pattern at this river basin. The results of the present study also show that the proposed IDNN achieves better accuracy than the MLP-NN and RBFNN for the extreme rainfall pattern events; however, the IDNN is still less accurate if compared with the forecasting accuracy for non-extreme events. In addition, the proposed IDNN showed better accuracy for mimicking the consecutive rainfall pattern than the classical MLP-NN and /or RBFNN model.

For future research in applying AI-based models, it is highly recommended to find a better and more reliable preprocessing method (rather than using the trial and error method) that can figure out the best input window size. In addition, it is recommended also to study the temporal dimension order before establishing the IDNN model configuration, in order to find the optimal input-delay length in model architecture. In this study we utilized the backpropagation optimization method, which may suffer from the problem of local optima. There are many advanced methods offered by researchers to overcome this drawback such as particle swarm optimization (PSO) and genetic algorithm (GA). For future research, such PSO and GA optimization methods could be re-adjusted within different types of training algorithms in the IDNN model in order to avoid a local optima problem and then enhance the overall forecasting accuracy.

Acknowledgements. This research is supported by: (1) a research grants to the first author by Smart Engineering System, University Kebangsaan Malaysia; and eScience Fund project 01-01-02-SF0581, ministry of Science, Technology and Innovation, (MOSTI) and UKM-DLP-2011-002 (2) a research grant to the second author from the Natural Science and Engineering Research

Council (NSERC) of Canada. The authors appreciate the support from Klang Gate Dam and Department of Drainage and Irrigation (DID), Malaysia for providing all the required data used for developing this research.

Edited by: D. Solomatine

References

Abrahart, R. J. and See, L. M.: Neural network modelling of non-linear hydrological relationships, Hydrol. Earth Syst. Sci., 11, 1563–1579, doi:10.5194/hess-11-1563-2007, 2007.

Akhtar, M. K., Corzo, G. A., van Andel, S. J., and Jonoski, A.: River flow forecasting with artificial neural networks using satellite observed precipitation pre-processed with flow length and travel time information: case study of the Ganges river basin, Hydrol. Earth Syst. Sci., 13, 1607–1618, doi:10.5194/hess-13-1607-2009, 2009.

Altunkaynak, A., Özger, M., and Çakmakçi, M.: Fuzzy Logic Modeling of the Dissolved Oxygen Fluctuations in Golden Horn, Ecol. Model., 189, 436–446, 2005a.

Altunkaynak, A., Özger, M., and Çakmakçi, M.: Water Consumption Prediction of Istanbul City by Using Fuzzy Logic Approach, Water Resour. Mange., 19, 641–654, 2005b.

Alvisi, S., Mascellani, G., Franchini, M., and Bárdossy, A.: Water level forecasting through fuzzy logic and artificial neural network approaches, Hydrol. Earth Syst. Sci., 10, 1–17, doi:10.5194/hess-10-1-2006, 2006.

Anctil, F. and Lauzon, N.: Generalisation for neural networks through data sampling and training procedures, with applications to streamflow predictions, Hydrol. Earth Syst. Sci., 8, 940–958, doi:10.5194/hess-8-940-2004, 2004.

Awwad, H., Valdes, J., and Restrepo, P.: Streamflow forecasting for Han River Basin, Korea, J. Water Resources Planning Management, 120, 651–673, 1994.

Beale, H. and Demuth, H. B.: Neural Network Toolbox for Use with MATLAB, 1st Edn., International Thomson Publishing, MA, USA, 2001.

Bishop, C. M.: Neural networks for pattern recognition, 1st Edn., Oxford University Press, UK, 1996.

Boucher, M.-A., Laliberté, J.-P., and Anctil, F.: An experiment on the evolution of an ensemble of neural networks for streamflow forecasting, Hydrol. Earth Syst. Sci., 14, 603–612, doi:10.5194/hess-14-603-2010, 2010.

Bowden, G. J., Dandy, G. C., and Maier, H. R.: Input determination for neural network models in water resources applications. Part 1 – Background and methodology, J. Hydrol., 301, 75–92, 2005.

Box, G., H. and Jenkins, G.: Time Series Analysis: Forecasting and Control Holden-Day, San Francisco, 1970.

Bras, R. and Rodriguez-Iturbe, I.: Random Functions and Hydrology. , Reading, MA, Addison-Wesley, 1985.

Campolo, M., Andreussi, P., and Soldati, A.: River flood forecasting with neural network model, Water Resour. Res., 35, 1191–1197, 1999.

Cheng, Y.: Evaluating an autoregressive model for stream flow forecasting, in: Proc. Hydraulic Eng. Conf., 1105–1109, 1994.

Chiu, C. L. (Ed.): Applications of Kalman filter to hydrology, hydraulics, and water resources. Proceedings of AGU Chapman Conference, Pittsburgh, Dept. of Civil Engineering, University of Pittsburgh Press, 1978.

Cinar, O., Hasar, H., and Kinaci, C.: Modeling of submerged membrane bioreactor treating cheese whey wastewater by artificial neural network, J. Biotech., 123, 204–209, 2006.

Coulibaly, P., Anctil, F., and Bobée, B.: Daily reservoir inflow forecasting using artificial neural networks with stopped training approach, J. Hydrol., 230, 244–257, 2000a.

Coulibaly, P., Anctil, F., and Bobée, B.: Neural network-based long-term hydropower forecasting system, Comp. Aided Civ. Infrastruct. Engrg., 15, 355–364, 2000b.

Coulibaly, P., Anctil, F., and Bobée, B.: Multivariate Reservoir Inflow Forecasting Using Temporal Neural Networks, ASCE J. Hydrol. Eng., 65, 367–376, 2001.

de Vos, N. J. and Rientjes, T. H. M.: Constraints of artificial neural networks for rainfall-runoff modelling: trade-offs in hydrological state representation and model evaluation, Hydrol. Earth Syst. Sci., 9, 111–126, doi:10.5194/hess-9-111-2005, 2005.

El-Shafie, A. and Noureldin, A.: Generalized versus non-generalized neural network model for multi-lead inflow forecasting at Aswan High Dam, Hydrol. Earth Syst. Sci., 15, 841–858, doi:10.5194/hess-15-841-2011, 2011.

El-Shafie, A., Mukhlisin, M., Najah, A. A., and Taha M. R.: Performance of artificial neural network and regression techniques for rainfall-runoff prediction, International Journal of the Physical Sciences, 6, 1997–2003, 2011a.

El-Shafie, A., Jaafer, O., and Seyed, A.: Adaptive neuro-fuzzy inference system based model for rainfall forecasting in Klang River, Malaysia, Int. J. Phys. Sci., 6, 2875–2888, 2011b.

El-Shafie, A. H., El-Shafie, A., El Mazoghi, H. G., Shehata, A., and Taha, M.R.: Artificial neural network technique for rainfall forecasting applied to Alexandria, Egypt, Int. J. Phys. Sci., 6, 1306–1316, 2011c.

Fernando, T. M. K. G., Maier, H. R., Dandy, G. C., and May, R. J.: Efficient selection of inputs for artificial neural network models, Proc. of MODSIM 2005 International Congress on Modelling and Simulation: Modelling and Simulation Society of Australia and New Zealand, December 2005, edited by: Zerger, A. and Argent, R. M., 1806–1812, 2005.

Gibson, K. and Dodge, R. A.: Hydraulic model studies of modification to Klang Gate Dam, Malaysia, Technical report, Engineering and Research Center, Denver, Colorado, 1983.

Hagan, M. and Menhaj, M.: Training Feedforward Networks with the Marquardt Algorithm, IEEE T. Neural Networ., 5, 989–993, 1994

Haykin, S.: Neural Networks: A Comprehensive Foundation, Macmillan College Publishing Company, New York, 1994.

Haykin, S.: Neural networks: Comprehensive foundation, Prentice-Hall, Upper Saddle River, N. J., 1999.

Hiew, K. L.: Flood mitigation and flood risk management in Malaysia, Journal of Floodplain Risk Management, 11, 1205–1216, 1996.

Hsu, K. L., Gupta, H. V., and Sorooshian, S.: Artificial neural network modeling of rainfall-rainoff process, Water Resour. Res., 31, 2517–2530, 1995.

Hung, N. Q., Babel, M. S., Weesakul, S., and Tripathi, N. K.: An artificial neural network model for rainfall forecasting in Bangkok, Thailand, Hydrol. Earth Syst. Sci., 13, 1413–1425, doi:10.5194/hess-13-1413-2009, 2009.

Jain, S. K., Das, D., and Srivastava, D. K.: Application of ANN for reservoir inflow prediction and operation, J. Water Resour. Plann. Manage., ASCE, 125, 263–271, 1999.

Lapedes, A. and Farber, R.: Nonlinear signal processing using neural networks, Technical Report of Los Alamos National Laboratory, LAUR- 87-2662, 1987.

Modarres, R.: Multi-criteria validation of artificial neural network rainfall-runoff modeling, Hydrol. Earth Syst. Sci., 13, 411–421, doi:10.5194/hess-13-411-2009, 2009.

Najah, A., El-Shafie, A., Karim, O. A., and Jaafar, O.: Water Quality Prediction Model Utilizing Integrated Wavelet-ANFIS Model with Cross Validation, Neural. Comput. Appl., in press, doi:10.1007/s00521-010-0486-1, 2010a.

Najah, A., Elshafie, A., Karim, O. A., and Jaffar, O.: Evaluation the efficiency of radial basis function neural network for prediction of water quality parameters, Eng. Int. Syst., 4, 221–231, 2010b.

Najah, A., El-Shafie, A., Karim, O. A., and Jaafar, O.: Integrated versus isolated scenario for prediction dissolved oxygen at progression of water quality monitoring stations, Hydrol. Earth Syst. Sci., 15, 2693–2708, doi:10.5194/hess-15-2693-2011, 2011.

Narendra, K. and Parthasarathy, K.: Identification and control of dynamical systems using neural networks, Trans. Neural Networks, 1, 4–27, 1990.

Noureldin, A., El-Shafie, A., and Taha, M. R.: Optimizing Neuro-fuzzy Modules for Data Fusion of Vehicular Navigation Systems Using Temporal Cross-validation, Engineering Applications of Artificial Intelligence, 49–61, February, 2007.

Noureldin, A., El-Shafie, A., and Bayoumi, A.: GPS/INS Integration Utilizing Dynamic Neural Network for Vehicular Navigation, Information Fusion, Elsevier, 11, 317–327, 2011.

Orr, M. J. L.: Introduction to radial basis function networks, Scotland, University of Edinburgh, 1996.

Ripley, B. D.: Pattern Recognition and Neural Networks. 1st edition, Cambridge University Press, NY, USA, 1996.

Saad, M., Bigras, P., Turgeon, A., and Duquette, R.: Fuzzy learning decomposition for scheduling of hydroelectric power systems Water Resour. Res., 32, 179–186, 1996.

Sajikumar, N. and Thandaveswara, B. S.: Non-linear rainfall runoff model using artificial neural network J. Hydrol., 216, 32–35, 1999.

Tan, S. S.: Optimizing the operation release policy for Klang Gate Dam. Thesis of Bachelor, Faculty of Civil Engineering, Universiti Kebangsaan Malaysia, 2009.

Thirumalaiah, K. and Deo, M. C.: Hydrological forecasting using neural networks, J. Hydrol. Eng., ASCE, 5, 180–189, 2000.

Tokar, A. S. and Johnson, P. A.: Rainfall-runoff modeling using artificial neural networks, J. Hydrol. Eng., ASCE, 4, 232–239, 1999.

Toth, E., Montanari, A., and Brath, A.: Comparison of short-term rainfall prediction model for real-time flood forecasting, J. Hydrol., 239, 132–147, 2000.

Weigend, S., Mangeas, M., and Srivastava, N.: Nonlinear gated experts for time series: discovering regimes and avoiding overfitting, Int. J. Neural Syst., 6, 373–99, 1995.

Xiong, L., O'Connor, K. M., and Guo, S.: Comparison of three updating schemes using artificial neural network in flow forecasting, Hydrol. Earth Syst. Sci., 8, 247–255, doi:10.5194/hess-8-247-2004, 2004.

Zealand, C. M., Burn, D. H., and Simonovic, S. P.: Short term stream flow forecasting using artificial neural networks, J. Hydrol., 214, 32–48, 1999.

Zhang, G., Patuwo, B., and Hu, M.: Forecasting with artificial neural networks: The state of the art, Int. J. Forecast., 14, 35–62, 1998.