Reinforced Two-Step-Ahead Weight Adjustment Technique for Online Training of Recurrent Neural Networks

Li-Chiu Chang, Pin-An Chen, and Fi-John Chang

***Abstract*— A reliable forecast of future events possesses great value. The main purpose of this paper is to propose an innovative learning technique for reinforcing the accuracy of two-step-ahead (2SA) forecasts. The real-time recurrent learning (RTRL) algorithm for recurrent neural networks (RNNs) can effectively model the dynamics of complex processes and has been used successfully in one-step-ahead forecasts for various time series. A reinforced RTRL algorithm for 2SA forecasts using RNNs is proposed in this paper, and its performance is investigated by two famous benchmark time series and a streamflow during flood events in Taiwan. Results demonstrate that the proposed reinforced 2SA RTRL algorithm for RNNs can adequately forecast the benchmark (theoretical) time series, significantly improve the accuracy of flood forecasts, and effectively reduce time-lag effects.**

***Index Terms*— Real-time recurrent learning (RTRL) algorithm, recurrent neural network (RNN), streamflow forecast, time series forecast.**

I. Introduction

MOST observational disciplines tend to infer properties of an uncertain system from the analysis of its measured data. The analytical technologies for extracting the meaningful characteristics of time series data have been widely discussed for a long time [1]. Many mature techniques associated with time series analysis were used in many important applications such as environment and marketing [2]. Because observations closer together in time generally would be more closely related than observations further apart, it is more difficult to obtain a satisfactory multistep-ahead forecast. The recursive use of one-step-ahead forecasts for many time steps into the future is a commonly used strategy, which unfortunately has been shown to have shortcomings in real-world applications [3]. This is mainly because a small forecast error at the beginning could propagate into the future. To solve such a problem, it is argued whether an iterative adjustment of the model's parameters based on additional information, such as antecedent observed values and/or model outputs, would be beneficial to multistep-ahead forecasts.

Over the last few decades, artificial neural networks (ANNs) have been recognized for modeling the underlying nonlinearities and complexities in artificial or physical systems. Many ANNs were developed to solve different problems, such as rainfall and streamflow forecasting [4]–[9], seismic [10], reservoir flood control [11], financial forecasts [12], sunspot activity [13], and many other disciplines for multistep-ahead forecasts [13]–[17]. Most of these applied with neural networks are classified into static neural networks and can simulate the short-term memory structures within processes, whereas the extraordinary time variation characteristics of time series might not be well retained.

Lately, recurrent neural networks (RNNs) have attracted much attention [18]–[24] for extracting dynamic time variation characteristics. Because of their dynamic nature, RNNs have been successfully applied to a wide variety of problems such as system identification [25]–[27], speech processing and plant control [28], and time series forecasting [29]–[33]. RNNs are capable of improving forecast accuracy [3], [34]–[36]. The training of an RNN, however, could be time consuming [13], [37], such as back-propagation through time (BPTT). BPTT, designed for training RNNs, can be derived by unfolding the temporal operation of the network into a multilayer feedforward network. The two familiar implementations of BPTT are the batch mode (epoch-wise BPTT) and real-time mode (truncated BPTT) [38]. A potential drawback of truncated BPTT is that the memory effects exceeding the truncation depth (duration) cannot be captured by RNNs.

The real-time recurrent learning (RTRL) algorithm, proposed by Williams and Zipser [39], is an effective and efficient algorithm for training recurrent networks, and its name is derived from the fact that real-time adjustments are made to the synaptic weights of an RNN. A number of previous studies demonstrated that the RTRL algorithm for RNNs is very effective in modeling the dynamics of complex processes and can provide accurate forecasts [40]–[42], while some studies further made efforts to reduce the time complexity of the RTRL algorithm [43]–[45].

Due to geophysical conditions, reservoirs in Taiwan are relatively small when considering the amount of water falling on watersheds during typhoon events. A controlled spillway is equipped with mechanical gates to control the water release

Manuscript received September 2, 2011; revised May 6, 2012; accepted May 10, 2012. Date of publication June 14, 2012; date of current version July 16, 2012. This work was supported in part by the National Science Council, Taiwan, under Grant 100-2313-B-002-011-MY3.

L.-C. Chang is with the Department of Water Resources and Environmental Engineering, Tamkang University, New Taipei City 25137, Taiwan (e-mail: changlc@mail.tku.edu.tw).

P.-A. Chen and F.-J. Chang are with the Department of Bioenvironmental Systems Engineering, National Taiwan University, Taipei 10617, Taiwan (e-mail: f98622003@ntu.edu.tw; changfj@ntu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2012.2200695

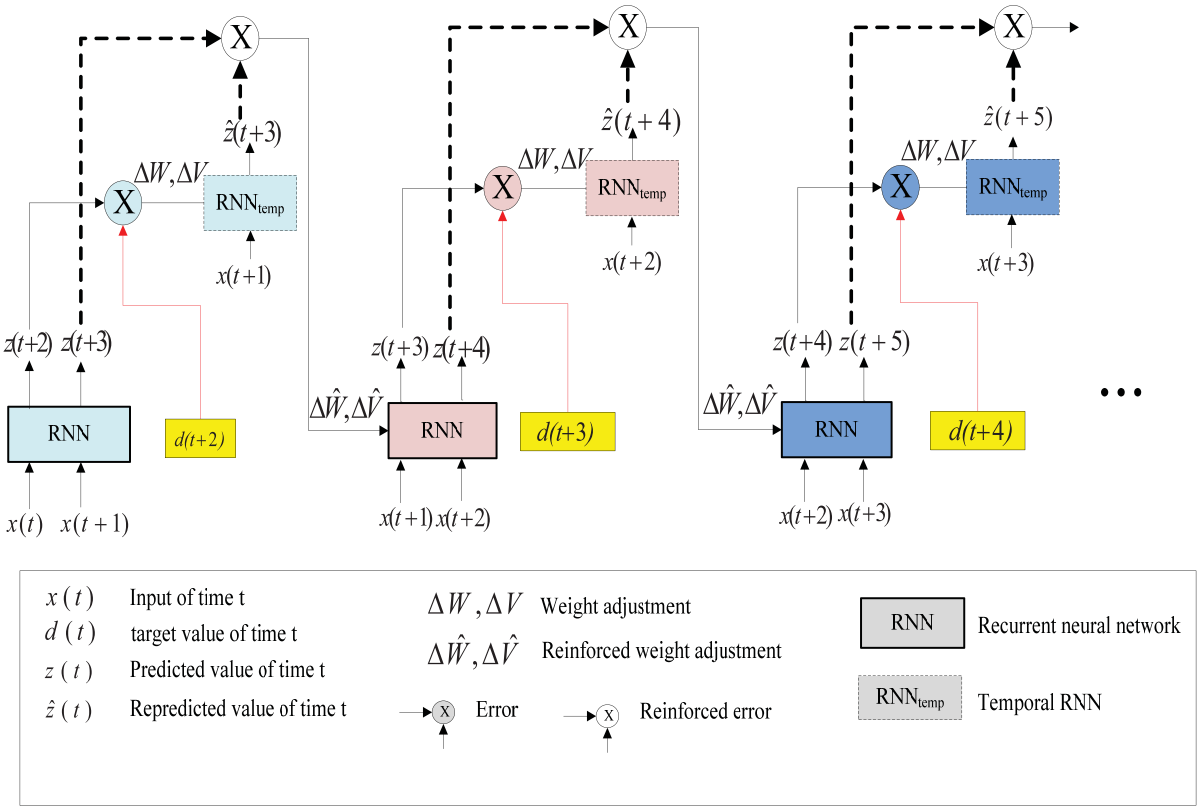

Fig. 1. Reinforced 2SA weight adjustment procedure for RNN.

of a dam. The discharge rate must be carefully arranged by suitably controlling the spillway gates. In this paper, it will take about two hours to fully open the floodgates. Consequently, an accurate two-step-ahead (2SA) forecast will be crucial and beneficial to decision makers. As a result, the major goal of this paper is to develop a reinforced 2SA RTRL (R-RTRL) algorithm for RNNs and further demonstrate its reliability and applicability in 2SA forecasts for two famous benchmark chaotic time series and an inflow of reservoir in Taiwan by mitigating time-lag effects and increasing the forecast accuracy. Its performance is compared with that of the back-propagation neural networks (BPNNs) and RNNs.

II. 2SA Forecast Methodology

A. Concept and Procedure

The basic idea for forecasts is usually the case that additional information from antecedent observed values and/or model outputs will be beneficial to forecasts, and forecast results tend to be closer to the true values as the forecast model can be iteratively adjusted through model performance with a reduction of errors. The purpose of this paper is to provide evidence for the aforementioned arguments and propose a reinforced weight adjustment technique for 2SA forecast models under RNN infrastructures for increasing forecast accuracy through incorporating the antecedent forecasted values and observed values into consecutive temp networks to adjust weights in the learning process. Fig. 1 shows the proposed reinforced 2SA weight adjustment procedure for online learning of RNNs. The proposed procedure is used to develop an RNN of the 2SA RTRL algorithm, and its applicability and effectiveness for various time series are shown in the following section.

B. 2SA RTRL Algorithm

An RTRL algorithm was derived from the fact that real-time adjustments are made to the synaptic weights of a fully connected RNN [39]. Based on the one-step-ahead RTRL algorithm, Chang *et al.* [46] devised a 2SA RTRL algorithm. Its weight adjustments could only be completed until obtaining the observed value at time $t+2$, which means the antecedent information of the observed $x(t+1)$ and model output $z(t+3)$ is ignored. Consequently, its weight adjustment process does not fully use the antecedent information and has time-lag problems. To fix this shortcoming, this paper develops a 2SA (R-RTRL) algorithm by utilizing all of the most current information.

As shown in Fig. 1, at time $t+2$, the weights are adjusted by the differences between observed and forecasted values. The RNN_{temp} with adjusted weights can be used to produce a new updating output $\hat{z}(t+3)$ at time $t+1$, and the reinforced error between the updating and and former forecasted outputs at time $t+1$ can then be utilized to reinforce the weight adjustments $\Delta \hat{W}(t)$ and $\Delta \hat{V}(t+1)$. The detailed algorithm is shown in the next section.

C. Deriving the Algorithm

Fig. 2 is a 2SA R-RTRL architecture with M external inputs and K outputs. Let $x(t)$ denote the $M \times 1$ input vector at

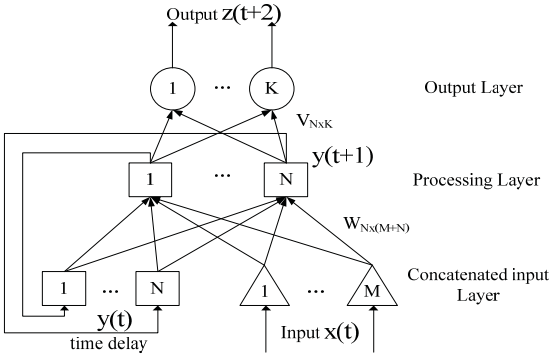

Fig. 2. Architecture of 2SA RNN.

discrete time t, $y(t+1)$ denote the corresponding $N\times 1$ vector one-step later at time $t+1$ in the processing layer, and $z(t+2)$ denote the corresponding $K \times 1$ output vector.

The $x(t)$ and $y(t)$ are concatenated to form the $(M+N)\times 1$ vector $\mu(t)$, whose ith element is denoted by $\mu_i(t)$. Let A denote the set of indices i, for which $x_i(t)$ is an external input, and B denote the set of indices i, for which $y_i(t)$ is the output of the processing layer. Thus, vector $\mu(t)$ can be represented as

$$\mu_i(t) = \begin{cases} x_i(t), & \text{if } i \in A \\ y_i(t), & \text{if } i \in B. \end{cases} \tag{1}$$

W and V denote the weight matrix in the processing layer and output later, respectively. $\boldsymbol{W} \leftrightarrow w_{ji}$ and $\boldsymbol{V} \leftrightarrow v_{kj}$ are of matrix forms. The output of neuron j in the processing layer is given by

$$y_j(t+1) = f\left(\text{net}_j(t+1)\right) = f\left(\sum_{i\in A\cup B} w_{ji}(t)\mu_i(t)\right). \tag{2}$$

The net output of neuron k in the output layer at time $t+2$ is computed by

$$z_k(t+2) = f\left(\text{net}_k(t+2)\right) = f\left(\sum_j v_{kj}(t+1)y_j(t+1)\right). \tag{3}$$

Let $d_k(t+2)$ denote the target value of neuron k at time $t+2$. The error vector is defined by

$$e_k(t+2) = d_k(t+2) - z_k(t+2). \tag{4}$$

Define the instantaneous overall network error at time $t+2$ as

$$E(t+2) = \frac{1}{2}\sum_{k=1}^{K} e_k^2(t+2). \tag{5}$$

To minimize (5), the weight adjustments can be computed as

$$\Delta w_{mn}(t) = \eta_2\left[\sum_{k=1}^{K} e_k(t+2)f'(\text{net}_k(t+2))v_{kj}(t+1)\right] \times \pi_{mn}^{j}(t+1) \tag{6}$$

$$\Delta v_{kj}(t+1) = \eta_1 e_k(t+2)f'(\text{net}_k(t+2))y_j(t+1) \tag{7}$$

$$\pi_{mn}^{j}(t+1) = f'(\text{net}_j(t+1))\left[\sum_{i\in B} w_{ji}(t)\pi_{mn}^{i}(t) + \delta_{mj}u_n(t)\right] \tag{8}$$

where η_1 and η_2 are the learning-rate parameters and π is the dynamics of the system with initial condition $\pi_{mn}^{j}(0) = 0$. δ_{mj} has a value of 1 if and only if $j = m$, otherwise its value is zero [46].

The weights are adjusted at time $t+2$; however, the information from time $t+1$ does not contribute to the weight adjustment. We believe the closest antecedent ($t+1$ in this case) information is crucial and could further diminish the time-lag effects. Consequently, we proposed that the adjusted weights should be fed back to the network at time $t+1$ through incorporating the closest antecedent information (i.e., the observed $x(t+1)$ and model output $z(t+3)$) to recalculate its forecasted value, and then the adjusted weights are further modified through the comparison of the original forecasted value $z_k(t+3)$ and the reforecasted value. $\hat{z}_k(t+3)$ is calculated by the following equations:

$$\hat{y}_j(t+2) = f(n\hat{e}t_j(t+2)) = f\left(\sum_{i\in A\cup B}(w_{ji}(t)+\Delta w_{ji}(t))\mu_i(t+1)\right) \tag{9}$$

$$\hat{z}_k(t+3) = f(n\hat{e}t_k(t+3)) = f\left(\sum_j (v_{kj}(t+1)+\Delta v_{kj}(t+1))\hat{y}_j(t+2)\right). \tag{10}$$

Then the reinforced error vector $\hat{e}_k(t+3)$ of neuron k is defined by

$$\hat{e}_k(t+3) = \hat{z}_k(t+3) - z_k(t+3). \tag{11}$$

Therefore, the instantaneous overall reinforced error is defined by

$$\hat{E}(t+3) = \frac{1}{2}\sum_{k=1}^{K}\hat{e}_k^2(t+3). \tag{12}$$

To minimize (12), the additional change of adjusted weights can be written as

$$\Delta\hat{v}_{kj} = -\eta_3\frac{\partial\hat{E}(t+3)}{\partial v_{kj}} \tag{13}$$

$$\Delta\hat{w}_{mn} = -\eta_4\frac{\partial\hat{E}(t+3)}{\partial w_{mn}} \tag{14}$$

where η_3 and η_4 are the learning-rate parameters.

Now

$$\frac{\partial\hat{E}(t+3)}{\partial v_{kj}} = \hat{e}_k(t+3)\left[f'\left(n\hat{e}t_k(t+3)\right)\left(1+\frac{\partial\left(\Delta v_{kj}\right)}{\partial v_{kj}}\right) \times\hat{y}_j(t+2) - f'\left(net_k(t+3)\right)y_j(t+2)\right] \tag{15}$$

$$\frac{\partial(\Delta v_{kj})}{\partial v_{kj}} = \eta_1\left\{-\left[f'(net_k(t+2))y_j(t+1)\right]^2 + e_k(t+2)f''(net_k(t+2))y_j^2(t+1)\right\} \tag{15.1}$$

$$\frac{\partial \hat{E}(t+3)}{\partial w_{mn}} = \sum_k \hat{e}_k(t+3)\Big\{ f'(n\hat{e}t_k(t+3)) \times \Big[(v_{kj}(t+1)+\Delta v_{kj})\frac{\partial \hat{y}_j(t+2)}{\partial w_{mn}}\Big] - f'(net_k(t+3))v_{kj}(t+1)\pi^j_{mn}(t+2)\Big\} \quad (16)$$

$$\frac{\partial \hat{y}_j(t+2)}{\partial w_{mn}} = f'(n\hat{e}t_j(t+2)) \times \Big\{ \sum_i \Big[(w_{ji}(t)+\Delta w_{ji})\frac{\partial y_i(t+1)}{\partial w_{mn}}\Big] + \Big(\delta_{mj}+\frac{\partial(\Delta w_{ji})}{\partial w_{mn}}\Big)\mu_n(t+1)\Big\} \quad (16.1)$$

$$\frac{\partial(\Delta w_{mn})}{\partial w_{mn}} = \eta_2 \sum_{k=1}^{K}\Big\{ -\Big[f'(net_k(t+2))v_{kj}(t+1)\frac{\partial y_j(t+1)}{\partial w_{mn}}\Big]^2 + e_k(t+2)f''(net_k(t+2))\Big[v_{kj}(t+1)\frac{\partial y_j(t+1)}{\partial w_{mn}}\Big]^2 + e_k(t+2)f'(net_k(t+2))v_{kj}(t+1)\frac{\partial^2 y_j(t+1)}{\partial w^2_{mn}}\Big\}. \quad (16.2)$$

The first term in (15) is the additional information produced by the reforecasted value, and the first term in (16) is the information produced by the original forecasted value $z_k(t+3)$.

It is assumed that the initial state of the network at time $t = 0$ has no functional dependence on the weights, and, therefore

$$\frac{\partial^2 y_j(0)}{\partial w^2_{mn}(0)} = 0 \quad (17)$$

$$\frac{\partial^2 y_j(t+1)}{\partial w^2_{mn}} = f''(net_j(t+1)) \times \Big[\sum_{i\in B} w_{ji}(t)\frac{\partial y_i(t)}{\partial w_{mn}} + \delta_{mj}\mu_n(t)\Big]^2 + f'(net_j(t+1))\Big[\delta_{mj}\frac{\partial y_n(t)}{\partial w_{mn}} + \sum_{i\in B} w_{ji}(t)\frac{\partial^2 y_i(t)}{\partial w^2_{mn}} + \delta_{mj}\frac{\partial y_n(t)}{\partial w_{mn}}\Big]. \quad (18)$$

We define a reinforced dynamic system described by a triple indexed set of variables $\{\lambda^j_{mn}(t)\}$, where $\lambda^j_{mn}(t) = [\partial^2 y_j(t)/\partial w^2_{mn}]$ for all $j \in B$, $m \in B$, and $n \in A \cup B$. For each time step t and all appropriate m, n, and j, the reinforced dynamics system is governed by

$$\lambda^j_{mn}(t+1) = f''(net_j(t+1))\Big[\sum_{i\in B} w_{ji}(t)\pi^i_{mn}(t) + \delta_{mj}\mu_n(t)\Big]^2 + f'(net_j(t+1))\Big[2\delta_{mj}\pi^n_{mn}(t) + \sum_{i\in B} w_{ji}(t)\lambda^i_{mn}(t)\Big] \quad (19)$$

with initial condition $\lambda^j_{mn}(0) = 0$. The weight adjustments of the R-RTRL algorithm for the RNN are then shown as follows:

$$w^{\text{new}}_{mn} = w_{mn} + \Delta w_{mn} + \Delta \hat{w}_{mn}$$
$$v^{\text{new}}_{kj} = v_{kj} + \Delta v_{kj} + \Delta \hat{v}_{kj}. \quad (20)$$

III. Models Tested and Verified by Synthetic Time Series

To demonstrate the proposed R-RTRL algorithm for the RNN can construct a robust 2SA predictor by effectively utilizing all of the most current information, another type of RNN which possesses a highly interconnected and recurrent topology of nonlinear processing elements, called echo state network (ESN) [47], [48], and one of the static-type NN, BPNNs with one-step-ahead and 2SA predictors (BPNN_{I} and BPNN_{II}) are trained for 2SA forecasts. The forecast ability of the proposed R-RTRL network is first compared on two simulated chaotic systems: Mackey-Glass and Lorenz time series. The forecasts on the future values of the chaotic Mackey-Glass and Lorenz time series are recognized as benchmark time series that have been commonly used and reported by a number of researchers when comparing the learning and generalization ability of different neural networks [49]–[52].

The Mackey-Glass differential delay equation is defined as follows:

$$\frac{\dot{dx}(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \quad (21)$$

where the initial condition $x(0) = 1.2$, $\tau = 17$.

A total of 1000 input-output data pairs were extracted where the first 500 pairs were used for training while the remaining 500 pairs were used for testing.

The Lorenz series is given by

$$\frac{dx(t)}{dt} = \sigma\,[y(t)-x(t)]$$
$$\frac{dy(t)}{dt} = x(t)\,[r-z(t)] - y(t)$$
$$\frac{dz(t)}{dt} = x(t)y(t) - bz(t). \quad (22)$$

A time series of x with variable length equal to 2500 was generated by (22) with parameter values $\sigma = 10$, $r = 28$, and $b = 8/3$. The first 1500 samples were used for training while the remaining 1000 samples were used to test the models. Based on Takens' [53] embedding theorem, a series of observations from a chaotic system can reconstruct an attractor that is a subset of the phase space of the system with two parameters of time delay (T) and dimension (D). If appropriate time delay and dimension are selected, the attractor will retain the topological properties and reveal the hidden information of the original dynamic system. The mutual information method [54] and Cao's method [55] are used to determine proper time delay (T) and dimension (D), respectively. Therefore, the embedding dimension $D = 3$ and time delay $T = 7$ are determined for the Mackey-Glass time series, while $D = 4$ and $T = 6$ are for the Lorenz time series.

TABLE I
MODEL PERFORMANCE OF 2SA FORECASTING FOR MACKEY-GLASS TIME SERIES

	Training				Testing				
	RMSE	**NMSE**	G_{bench}	R_p	**RMSE**	**NMSE**	G_{bench}	G_{bench}**II**	R_p
R-RTRL	3.56E-03	2.47E-04	0.997	48.6	3.51E-03	2.41E-04	0.997	–	48.7
RTRL	6.60E-03	8.49E-04	0.990	43.2	6.49E-03	8.26E-04	0.990	−2.4	43.4
ESN	2.67E-04	1.37E-06	0.999	71.1	5.29E-03	5.48E-04	0.994	−1.3	45.2
BPNN$_{\text{I}}$	1.18E-02	2.70E-03	0.968	38.2	1.17E-02	2.68E-03	0.968	−10.1	38.3
BPNN$_{\text{II}}$	9.09E-03	1.61E-03	0.981	40.4	9.05E-03	1.61E-03	0.981	−5.7	40.5

TABLE II
MODEL PERFORMANCE OF 2SA FORECASTING FOR LORENZ TIME SERIES

	Training				Testing				
	RMSE	**NMSE**	G_{bench}	R_p	**RMSE**	**NMSE**	G_{bench}	G_{bench}**II**	R_p
R-RTRL	3.26E-02	2.08E-05	0.998	47.7	3.82E-02	2.38E-05	0.998	–	46.3
RTRL	5.49E-02	5.91E-05	0.995	42.5	6.25E-02	6.38E-05	0.995	−2.7	40.5
ESN	1.06E-04	2.21E-10	0.999	97.5	4.76E-02	3.70E-05	0.997	−0.56	44.3
BPNN$_{\text{I}}$	1.41E-01	3.88E-04	0.976	35.2	1.66E-01	4.43E-04	0.972	−15.7	34.3
BPNN$_{\text{II}}$	7.00E-02	9.59E-05	0.993	41.1	8.35E-02	1.14E-04	0.991	−3.8	39.5

Training datasets are used to construct neural network models. The 2SA forecast is performed for each case based on: 1) the constructed R-RTRL network; 2) the constructed RTRL network; 3) the ESN; 4) the constructed BPNN$_{\text{I}}$; and 5) the constructed BPNN$_{\text{II}}$. The root-mean-square error (RMSE), normalized mean-squared error (NMSE), prediction gain (R_p) [44], and the goodness-of-fit with respect to the benchmark (G_{bench}) [56] are used for comparison. The NMSE, R_p, and G_{bench} are defined as

$$NMSE = \frac{\sum_{i=1}^{n} (Q_i - \hat{Q}_i)^2}{\sum_{i=1}^{n} (Q_i - \overline{Q})^2} \tag{23}$$

$$R_p = 10 \log_{10} \left(\frac{\delta_x^2}{\delta_e^2} \right) [dB] \tag{24}$$

$$G_{\text{bench}} = 1 - \frac{\sum_{i=1}^{n} (Q_i - \hat{Q}_i)^2}{\sum_{i=1}^{n} (Q_i - Q_{i,\text{bench}})^2} \tag{25}$$

where Q_i is the observed value in the ith step, $\hat{Q}_i$ is the forecasted value in the ith step, $\overline{Q}$ represents the average of observed values, n is the number of data points, δ_x^2 is the mean-square value of the input signal, δ_e^2 is the mean-square value of the forecast error, and $Q_{i,\text{bench}}$ is the previous observed value, e.g., for the nth-step-ahead forecast, $Q_{i,\text{bench}} = Q_{i-n}$, and in this case, $n = 2$. Furthermore, to compare the proposed R-RTRL with the other models, the criterion G_{bench} II was defined, where the benchmark $Q_{i,\text{bench}}$ is the forecasted value of the proposed R-RTRL models in the ith step.

For comparison, the processing layers of the RTRL network and the proposed R-RTRL network consist of eight neurons for the Mackey-Glass time series and six neurons for the Lorenz time series, respectively, while the processing layers of the BPNN$_{\text{I}}$ and BPNN$_{\text{II}}$ consist of 12 neurons for the Mackey-Glass time series and six neurons for the Lorenz time series, respectively. And the number (N) of center vectors of ESN is 400 with 1% connectivity for both the Mackey-Glass and Lorenz time series. The numbers of the processing neurons and layers determined above are identified the best by a number of trials. Results are summarized in Tables I and II and presented in Figs. 3 and 4, which clearly demonstrate the following.

1) The BPNN$_{\text{I}}$ has the worst performance in both the Mackey-Glass and Lorenz time series. It reveals that small forecast errors that occured at the beginning of each recursion will accumulate and propagate to the future when recursively using the one-step-ahead forecast for the 2SA forecast, which will result in poor forecast accuracy.
2) The performance of dynamic neural networks (R-RTRL, RTRL, and ESN) is better than that of static neural networks (BPNN$_{\text{I}}$ and BPNN$_{\text{II}}$) in each case. It shows that the dynamic neural network can effectively extract the dynamic characteristics of time variation.
3) The ESN has the best performance in the training stage, whereas it cannot perform as well in the testing stage because of the problem posed by the linear regression algorithm.
4) The proposed R-RTRL algorithm has the best testing performance of all of the models in terms of all indexes (RMSE, NMSE, G_{bench}, and R_p values) in both cases. It demonstrates that the proposed algorithm that takes the closest antecedent information into consideration can effectively adjust synaptic weights, and the constructed network can provide reliable and accurate forecasts in the real-time 2SA forecast.

Closely assessing the results shown in Tables I and II, RTRL, ESN, BPNN$_{\text{I}}$, and BPNN$_{\text{II}}$ compared with the proposed R-RTRL networks have negative G_{bench}II values for both the Mackey-Glass time series and the Lorenz time series in the testing stage, which represents the outperformance of R-RTRL networks. Results indicate that using the closest

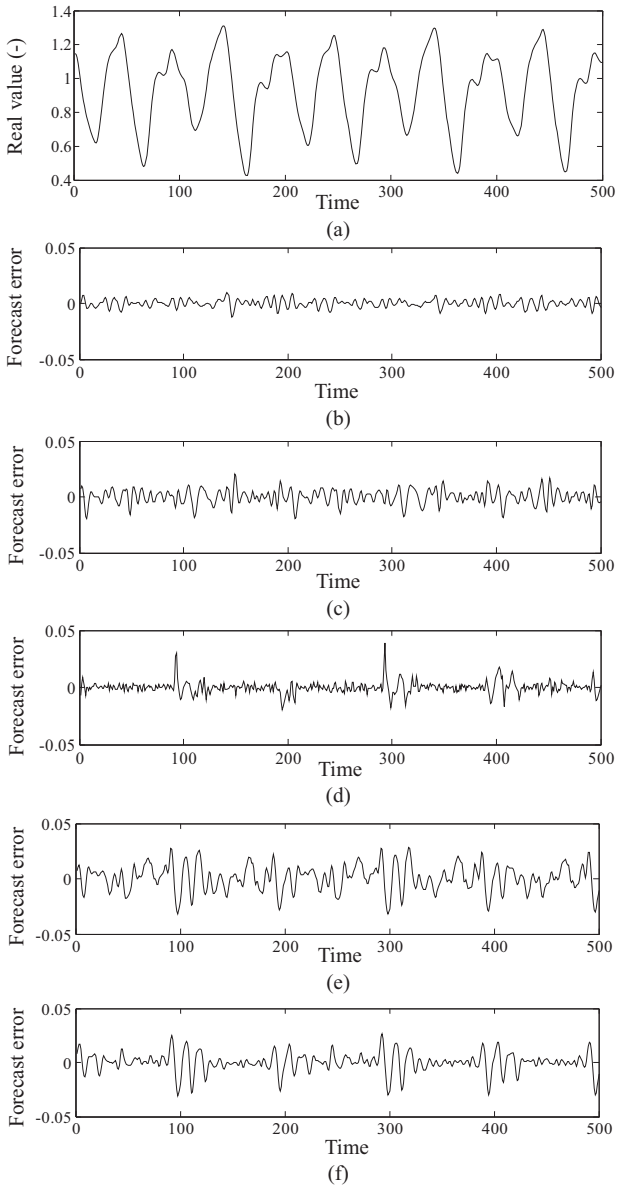

Fig. 3. (a) Mackey-Glass time-series (testing dataset). (b)–(f) 2SA forecast errors for R-RTRL, RTRL, ESN, $BPNN_I$, and $BPNN_{II}$, respectively.

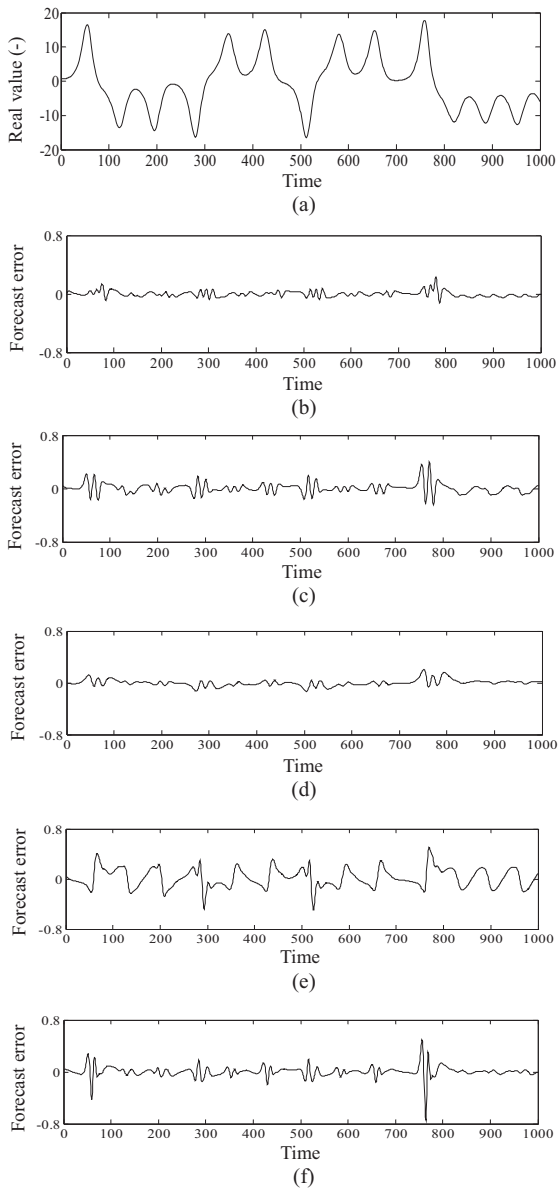

Fig. 4. (a) Lorenz time-series (testing data set). (b)–(f) 2SA forecast errors for R-RTRL, RTRL, ESN, BPNNI, and BPNNII, respectively.

antecedent information to iteratively adjust the weights is very useful and can significantly contribute to the accuracy of the model forecast. The relationship between run time per learning cycle and the number of neurons in the processing layers of the neural networks for the 2SA forecast of the Mackey-Glass time series when executing R-RTRL and RTRL algorithms on the same computer (Intel Core2 CPU Q8400 @ 2.66 GHz, 2.66 GHz, 3 GB of RAM) is calculated and compared in Fig. 5. Although the run time of the R-RTRL algorithm is longer than that of the RTRL algorithm, the run time of the R-RTRL algorithm is less than 1 s when the number of neurons increases to 10. It shows that the R-RTRL algorithm that additionally reinforces weight adjustments does not intensively increase the computation time or cause a barrier in practical applications. Also, the minimum time scale of the flood forecasting in this paper is one hour (while it could be days in other studies). Therefore, the additional time consumed of the proposed R-RTRL algorithm is satisfactorily acceptable when considering the significant improvement of forecast accuracy at the same time. As a result, we can conclude that the proposed R-RTRL network has effectively captured the most current and essential components of underlying dynamics and can be implemented to configure nonlinear dynamic systems such as chaotic time series.

IV. Applications

Flood forecasting not only represents a very complex nonlinear problem, but also is extremely difficult to model. Building a real-time stream-flow forecast model has always been one of the most important and challenging tasks for engineers. Taiwan, located in the subtropical zone of the North Pacific Ocean, is an island with mountainous terrains and steep landforms, where typhoons, usually coupled with heavy rainfalls, may cause downstream flooding within a

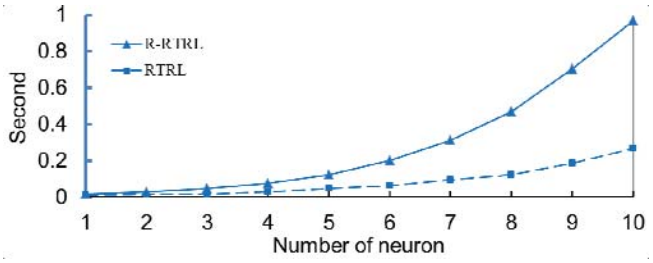

Fig. 5. Run time per learning cycle versus the number of neurons for 2SA forecasts.

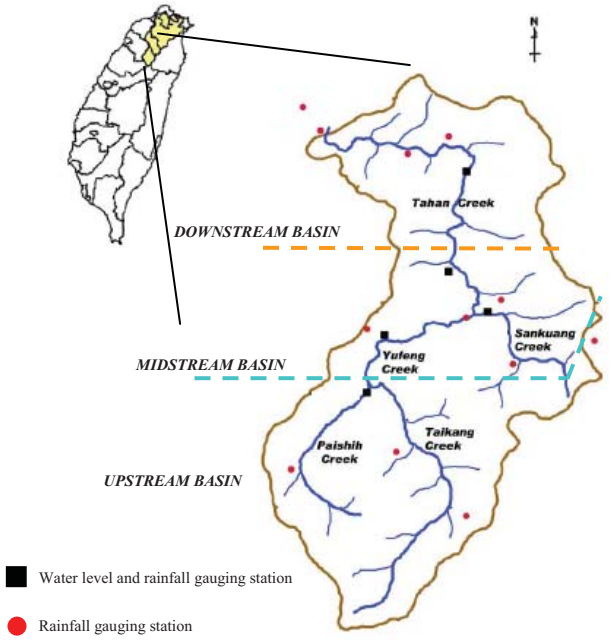

Fig. 6. Location of the Shihmen Reservoir and rainfall gauging stations.

few hours. An accurate inflow forecast is crucial for flood control and water resources management. The aforementioned R-RTRL network is applied to the Shihmen Reservoir for forecasting real-time inflow and is also compared with the other four models: RTRL network, ESN, $BPNN_I$, and $BPNN_{II}$.

The Shihmen Reservoir, situated upstream of the Tahan Creek in Northern Taiwan, operates for multiple purposes, such as water supply, flood control, hydropower generation, and recreation. The Shihmen Reservoir is the most important water resources facility for Taipei metropolitan areas and has an effective storage of about 2.35×10^8 m^3.

When a typhoon is expected, the water level in the dam can be reduced in advance to increase its floodwater storage capacity for future use. The concrete ogee-type spillway has six floodgates, where each gate is of 14×10.61 m, with the minimum discharge rate of 600 m^3/s and the maximum discharge rate of up to 11 400 m^3/s. It takes about two hours to fully open all the six floodgates; consequently an accurate two-hour-ahead (2SA) forecasting will be very beneficial to decision makers.

Locations of the basin and 16 rainfall gauging stations in this paper are shown in Fig. 6. The hourly inflow and precipitation data collected from 2001 to 2006 are available.

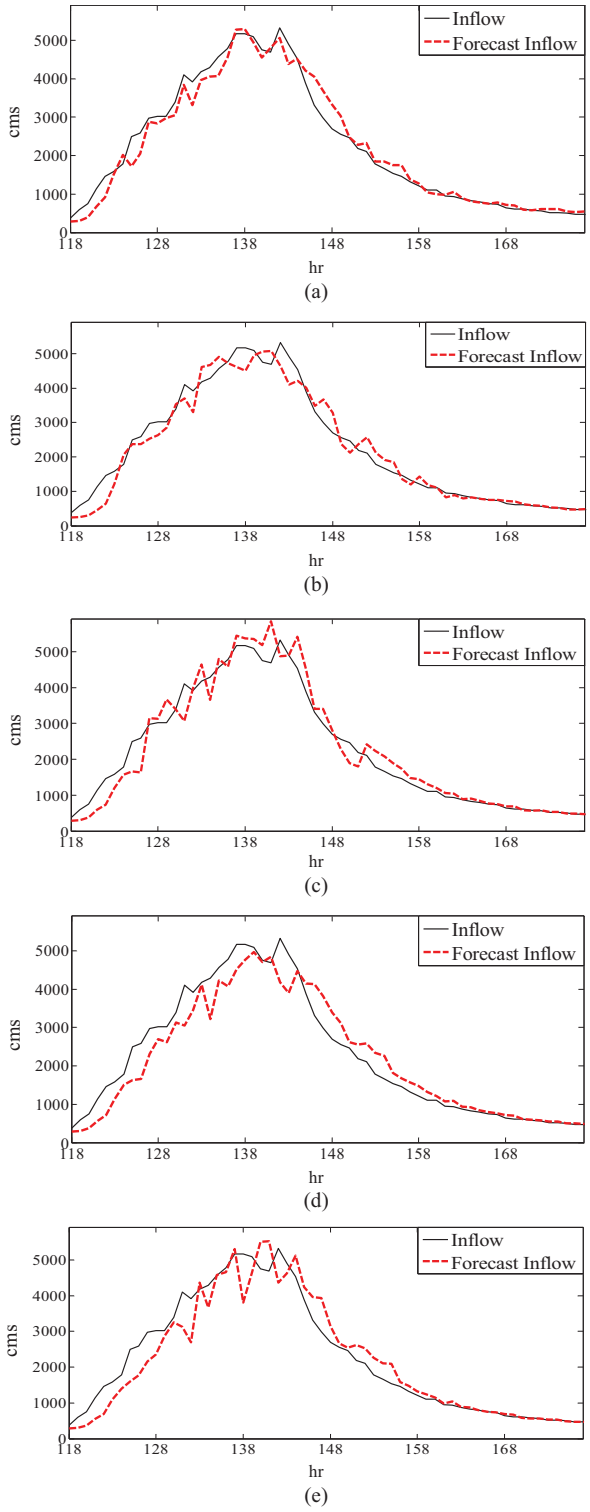

Fig. 7. 2SA stream-flow forecast (Matsa typhoon event of testing data set) based on (a) R-RTRL, (b) RTRL, (c) ESN, (d) BPNNI, and (e) BPNNII, respectively.

A total of 1492 datasets were used in this paper after eliminating the low-value stream-flow data of the year 2003. A total of 917 data were used for training while the remaining 575 data were used to test the models. Moreover, the Shihmen basin is divided into three areas: upstream basin, midstream basin, and downstream basin. Because the travel distance and

TABLE III
PERFORMANCE OF R-RTRL AND OTHER MODELS FOR FORECASTING RESERVOIR INFLOW

	Training				Testing				
	RMSE	**NMSE**	G_{bench}	R_p	**RMSE**	**NMSE**	G_{bench}	G_{bench}II	R_p
R-RTRL	147.03	1.91E-02	0.67	18.9	152.57	2.95E-02	0.55	–	17.4
RTRL	166.38	2.44E-02	0.58	17.8	204.74	5.31E-02	0.18	−0.8	14.9
ESN	123.52	1.50E-02	0.69	19.8	199.78	5.05E-02	0.22	−0.7	15.1
BPNN$_{\text{I}}$	217.54	4.17E-02	0.28	15.5	233.93	6.93E-02	−6.4E-02	−1.3	13.7
BPNN$_{\text{II}}$	179.42	2.84E-02	0.51	17.2	227.28	6.54E-02	−4.5E-03	−1.2	14.0

the transit time of flows moving from each station to the Shihmen Reservoir are different, the Kendall tau rank correlation coefficient [57] is applied to determine the highest correlation between the different time lags between rainfall and the inflow of the reservoir. The time lags of rainfall from the inflow of the reservoir to the downstream, midstream, and upstream basins are set to 6 h, 6 h, and 7 h, respectively, due to the highest value of the Kendall tau rank correlation coefficient. The input layers of the five different network models are established, based on stream-flow data together with rainfall data collected from three rainfall gauging stations in different areas, and each processing layer of its corresponding network consists of eight neurons, except the ESN model, which has $N = 25$ with 1% connectivity in the center vectors. The performance of these five models is presented in terms of the criteria of RMSE, NMSE, G_{bench}, R_p, and G_{bench}II.

Summarized results are presented in Table III. Results show that the proposed R-RTRL algorithm has smaller RMSE and NMSE values and much higher G_{bench} values than the other four network models in the testing stage. The proposed R-RTRL algorithm has similar results in the training and testing stages, while the other four models perform well only in the training stage, especially the ESN model. It shows the impressive stability and generalizability of the model based on the R-RTRL algorithm. Besides, the RTRL, ESN, and BPNN models can only produce small G_{bench} values (even negative numbers) in testing datasets, whereas the proposed algorithm produces high G_{bench} values (higher than 0.55) in both training and testing datasets. When closely assessing the results shown in Table III, RTRL, ESN, BPNN$_{\text{I}}$, and BPNN$_{\text{II}}$ compared with the proposed R-RTRL networks all have negative G_{bench}II for forecasting the 2SA stream flow in the testing stage. Apparently, the proposed R-RTRL network has much better performance than the other four networks.

To easily distinguish the performance of the five models, The Matsa typhoon event with the highest-peak-flow (above 5000 cm) data in the testing dataset were extracted to show the observed and forecasted flood flows from the proposed R-RTRL algorithm and the other four models in Fig. 7. Results demonstrate that the proposed algorithm can forecast 2SA stream-flow values well, whereas the other four models have significant time-lag phenomena and serious vibrations, which show that they cannot forecast 2SA stream flow values well. It appears that the proposed method could much more closely keep the flow trails, significantly diminish the time-lag effects, and provide much better and accurate 2SA forecasts.

V. CONCLUSION

A reliable and precise forecast of future events can be very beneficial. The RTRL networks can effectively model dynamic complex systems with high accuracy for one-step-ahead forecasts. Due to the lack of measurements in the forecasts horizon to adjust the parameters of the models in real time, a more than one-step-ahead forecast is more difficult to achieve satisfactorily and practically. Many engineering problems, however, require models providing multistep-ahead forecasts. In this paper, a 2SA flood forecast was crucial and beneficial to decision makers.

In this paper, a novel R-RTRL algorithm was derived for the RNN, which iteratively accommodated the one- and 2SA forecasted values and observed records in time series for diminishing the time-lag effects and increasing the accuracy in 2SA forecasts. Its ability of effective learning and accurate forecasting was demonstrated through two benchmark time series and the flood series of the Shihmen Reservoir in Northern Taiwan. For comparison, the original RTRL algorithm for RNN, ESN, BPNN$_{\text{I}}$, and BPNN$_{\text{II}}$ were also performed. In the cases of two benchmark time series, results indicated that the proposed R-RTRL network had better performance than the other four network models for the 2SA forecast in all cases, while the BPNN$_{\text{I}}$ had the worst performance. It revealed that small forecast errors that occurred at the beginning would accumulate and propagate to the future, which would decrease the 2SA forecast accuracy. For building a rainfall-runoff model for the Shihmen Reservoir, the proposed R-RTRL network had the best performance on the 2SA flood forecast, which significantly reduced the time-lag effects. The ESN model did not perform as well in the testing stage as in the training stage because of the problem posed by the linear regression algorithm, which means the ESN needed some methods, such as regularization method, to increase generalizability [33], [58], [59]. Moreover, the R-RTRL algorithm equipped with additional time-lag weight adjustments just slightly increased the computation complexity, which did not cause a barrier in practical applications. This paper demonstrated that the developed reinforced 2SA weight adjustment technique had great practicability and high accuracy for real-time 2SA forecasting by incorporating antecedent forecast information into the online learning process. And the proposed reinforced 2SA weight adjustment technique was suitable to be applied to neural networks, where online learning was required and could be further explored. Our future work will also involve a reinforced multistep-ahead online learning strategy.

ACKNOWLEDGMENT

The authors would like to thank the Editor, the Associate Editor, and the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

[1] G. Box and G. Jenkins, *Time Series Analysis: Forecasting and Control*. San Francisco, CA: Holden-Day, 1970.

[2] P. J. Brockwell and R. A. Davis, *Time Series: Theory and Methods*. New York: Springer-Verlag, 1987.

[3] A. G. Parlos, O. T. Rais, and A. F. Atiya, "Multi-step-ahead prediction using dynamic recurrent neural networks," *Neural Netw.*, vol. 13, no. 7, pp. 765–786, 2000.

[4] K. L. Hsu, H. V. Gupta, X. Gao, and S. Sorooshian, "Estimation of physical variables from multichannel remotely sensed imagery using a neural network: Application to rainfall estimation," *Water Resour. Res.*, vol. 35, no. 5, pp. 1605–1618, 1999.

[5] Y. M. Chiang, F. J. Chang, B. J. D. Jou, and P. F. Lin, "Dynamic ANN for precipitation estimation and forecasting from radar observations," *J. Hydrol.*, vol. 334, nos. 1–2, pp. 250–261, 2007.

[6] B. Sivakumar, A. W. Jayawardena, and T. M. K. G. Fernando, "River flow forecasting: Use of phase-space reconstruction and artificial neural networks approaches," *J. Hydrol.*, vol. 265, nos. 1–4, pp. 225–245, 2002.

[7] T. S. Hu, K. C. Lam, and S. T. Ng, "A modified neural network for improving river flow prediction," *Hydrol. Sci. J.*, vol. 50, no. 2, pp. 1–318, 2005.

[8] L. C. Chang, F. J. Chang, and Y. P. Wang, "Auto-configuring radial basis function networks for chaotic time series and flood forecasting," *Hydrol. Processes*, vol. 23, no. 17, pp. 2450–2459, 2009.

[9] Y. H. Chen and F. J. Chang, "Evolutionary artificial neural networks for hydrological systems forecasting," *J. Hydrol.*, vol. 367, nos. 1–2, pp. 125–137, 2009.

[10] T. Kerh, Y. Chan, and D. Gunaratnam, "Treatment and assessment of nonlinear seismic data by a genetic algorithm based neural network model," *Int. J. Nonlinear Sci. Numer. Simul.*, vol. 10, no. 1, pp. 45–56, 2009.

[11] L. C. Chang, F. J. Chang, and H. C. Hsu, "Real-time reservoir operation for flood control using artificial intelligent techniques," *Int. J. Nonlinear Sci. Numer. Simul.*, vol. 11, no. 11, pp. 887–902, 2010.

[12] F. Virili and B. Freisleben, "Neural network model selection for financial time series prediction," *Comput. Stat.*, vol. 16, no. 3, pp. 451–463, 2001.

[13] J.-X. Xie, C.-T. Cheng, K.-W. Chau, and Y.-Z. Pei, "A hybrid adaptive time-delay neural network model for multi-step-ahead prediction of sunspot activity," *Int. J. Environ. Pollut.*, vol. 28, nos. 3–4, pp. 364–381, 2006.

[14] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Trans. Acoust., Speech Signal Process.*, vol. 37, no. 3, pp. 328–339, Mar. 1989.

[15] S. P. Day and M. R. Davenport, "Continuous-time temporal back-propagation with adaptable time delays," *IEEE Trans. Neural Netw.*, vol. 4, no. 2, pp. 348–354, Mar. 1993.

[16] A. Yazdizadeh and K. Khorasani, "Adaptive time delay neural network structures for nonlinear system identification," *Neurocomputing*, vol. 47, nos. 1–4, pp. 207–240, 2002.

[17] D. Shi, H. J. Zhang, and L. M. Yang, "Time-delay neural network for the prediction of carbonation tower's temperature," *IEEE Trans. Instrum. Meas.*, vol. 52, no. 4, pp. 1125–1128, Aug. 2003.

[18] P. Campolucci, A. Uncini, F. Piazza, and B. D. Rao, "On-line learning algorithms for locally recurrent neural networks," *IEEE Trans. Neural Netw.*, vol. 10, no. 2, pp. 253–271, Mar. 1999.

[19] S. M. Zhou and L. D. Xu, "A new type of recurrent fuzzy neural network for modeling dynamic systems," *Knowl.-Based Syst.*, vol. 14, nos. 5–6, pp. 243–251, 2001.

[20] G. Serpen and Y. Xu, "Simultaneous recurrent neural network trained with non-recurrent backpropagation algorithm for static optimisation," *Neural Comput. Appl.*, vol. 12, no. 1, pp. 1–9, 2003.

[21] M. Assaad, R. Boné, and H. Cardot, "Study of the behavior of a new boosting algorithm for recurrent neural networks," in *Artificial Neural Networks: Formal Models and Their Applications*, vol. 3697. New York: Springer-Verlag, 2005, pp. 169–174.

[22] Y. M. Chiang, L. C. Chang, M. J. Tsai, Y. F. Wang, and F. J. Chang, "Dynamic neural networks for real-time water level predictions of sewerage systems-covering gauged and ungauged sites," *Hydrol. Earth Syst. Sci.*, vol. 14, no. 7, pp. 1309–1319, 2010.

[23] Z. Haiquan, Z. Xiangping, and H. Zhengyou, "Low-complexity non-linear adaptive filter based on a pipelined bilinear recurrent neural network," *IEEE Trans. Neural Netw.*, vol. 22, no. 9, pp. 1494–1507, Sep. 2011.

[24] S. Qing, "Robust initialization of a Jordan network with recurrent constrained learning," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 2460–2473, Dec. 2011.

[25] Y. Zhang and J. Wang, "Recurrent neural networks for nonlinear output regulation," *Automatica*, vol. 37, no. 8, pp. 1161–1173, 2001.

[26] J. Wang, H. Peng, and J. Xiao, "Identification of dynamic systems using recurrent fuzzy wavelet network," in *Advances in Neural Networks*, vol. 3972. New York: Springer-Verlag, 2006, pp. 802–807.

[27] L. Boquete, L. M. Bergasa, R. Barea, R. García, and M. Mazo, "Using a new model of recurrent neural network for control," *Neural Process. Lett.*, vol. 13, no. 2, pp. 101–113, 2001.

[28] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1999.

[29] P. Coulibaly, F. Anctil, and B. Bobee, "Multivariate reservoir inflow forecasting using temporal neural networks," *J. Hydrol. Eng.*, vol. 6, no. 5, pp. 367–376, 2001.

[30] T. Cheng and J. Wang, "Application of a dynamic recurrent neural network in spatio-temporal forecasting," in *Information Fusion and Geographic Information Systems*. Berlin, Germany: Springer-Verlag, 2007, pp. 173–186.

[31] S. V. Barai, A. K. Dikshit, and S. Sharma, "Neural network models for air quality prediction: A comparative study," *Soft Comput. Ind. Appl.*, vol. 39, pp. 290–305, Aug. 2007.

[32] Y. Zhang, L. Xi, and J. Liu, *Transient Air-Fuel Ratio Estimation in Spark Ignition Engine Using Recurrent Neural Networks Knowledge-Based Intelligent Information and Engineering Systems* (Lecture Notes in Computer Science). Berlin, Germany: Springer-Verlag, 2010, pp. 240–246.

[33] D. T. Mirikitani and N. Nikolaev, "Recursive Bayesian recurrent neural networks for time-series modeling," *IEEE Trans. Neural Netw.*, vol. 21, no. 2, pp. 262–274, Feb. 2010.

[34] M. Duhoux, J. Suykens, B. De Moor, and J. Vandewalle, "Improved long-term temperature prediction by chaining of neural networks," *Int. J. Neural Syst.*, vol. 11, no. 1, pp. 1–10, 2001.

[35] R. Bone, M. Crucianu, and J. P. A. de Beauville, "Learning long-term dependencies by the selective addition of time-delayed connections to recurrent neural networks," *Neurocomputing*, vol. 48, nos. 1–4, pp. 251–266, 2002.

[36] S. Chtourou, M. Chtourou, and O. Hammami, "A hybrid approach for training recurrent neural networks: Application to multi-step-ahead prediction of noisy and large data sets," *Neural Comput. Appl.*, vol. 17, no. 3, pp. 245–254, 2008.

[37] Z. Ahmad and Z. Jie, "Improving long range prediction for nonlinear process modelling through combining multiple neural networks," in *Proc. Int. Conf. Control Appl.*, vol. 2. 2002, pp. 966–971.

[38] R. J. Williams and J. Peng, "An efficient gradient-based algorithm for on-line training of recurrent network trajectories," *Neural Comput.*, vol. 2, no. 4, pp. 490–501, 1990.

[39] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, no. 2, pp. 270–280, 1989.

[40] K. Hirasawa, J. Murata, H. Jinglu, and J. Chunzhi, "Universal learning network and its application to robust control," *IEEE Trans. Syst., Man, Cybern., Part B, Cybern.*, vol. 30, no. 3, pp. 419–430, Jun. 2000.

[41] C. Li, S. He, X. Liao, and J. Yu, "Using recurrent neural network for adaptive predistortion linearization of RF amplifiers," *Int. J. RF Microw. Comput.-Aided Eng.*, vol. 12, no. 1, pp. 125–130, Jan. 2002.

[42] F. J. Chang, L. C. Chang, and H. L. Huang, "Real-time recurrent learning neural network for stream-flow forecasting," *Hydrol. Processes*, vol. 16, no. 13, pp. 2577–2588, 2002.

[43] D. Zipser, "A subgrouping strategy that reduces complexity and speeds up learning in recurrent networks," *Neural Comput.*, vol. 1, no. 4, pp. 552–558, 1989.

[44] K. P. Unnikrishnan and K. P. Venugopal, "Alopex: A correlation-based learning algorithm for feedforward and recurrent neural networks," *Neural Comput.*, vol. 6, no. 3, pp. 469–490, 1994.

[45] S. Haykin and L. Liang, "Nonlinear adaptive prediction of nonstationary signals," *IEEE Trans. Signal Process.*, vol. 43, no. 2, pp. 526–535, Feb. 1995.

[46] L. C. Chang, F. J. Chang, and Y. M. Chiang, "A two-step-ahead recurrent neural network for stream-flow forecasting," *Hydrol. Processes*, vol. 18, no. 1, pp. 81–92, 2004.
[47] H. Jaeger, "The echo state approach to analyzing and training recurrent neural networks," German National Research Center Information Technology, St. Augustin, Germany, Tech. Rep. GMD-148, 2001.
[48] R. Sacchi, M. C. Ozturk, J. C. Principe, A. A. F. Carneiro, and I. N. Silva, "Water inflow forecasting using the echo state network: A Brazilian case study," in *Proc. Int. Joint Conf. Neural Netw.*, 2007, pp. 2403–2408.
[49] J. S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man Cybern.*, vol. 23, no. 3, pp. 665–685, May–Jun. 1993.
[50] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Comput.*, vol. 1, no. 2, pp. 281–294, 1989.
[51] A. Gholipour, B. N. Araabi, and C. Lucas, "Predicting chaotic time series using neural and neurofuzzy models: A comparative study," *Neural Process. Lett.*, vol. 24, no. 3, pp. 217–239, 2006.
[52] M. Ardalani-Farsa and S. Zolfaghari, "Chaotic time series prediction with residual analysis method using hybrid Elman-NARX neural networks," *Neurocomputing*, vol. 73, nos. 13–15, pp. 2540–2553, 2010.
[53] F. Takens, "Detecting strange attractors in turbulence," *Dynamical Syst. Turbul.*, vol. 898, no. 1, pp. 366–381, 1981.
[54] A. M. Fraser and H. L. Swinney, "Independent coordinates for strange attractors from mutual information," *Phys. Rev. A*, vol. 33, no. 2, pp. 1134–1140, 1986.
[55] L. Cao, "Practical method for determining the minimum embedding dimension of a scalar time series," *Phys. D*, vol. 110, nos. 1–2, pp. 43–50, 1997.
[56] J. E. Nash and J. V. Sutcliffe, "River flow forecasting through conceptual models part I—A discussion of principles," *J. Hydrol.*, vol. 10, no. 3, pp. 282–290, 1970.
[57] D. R. Maidment, *Handbook of Hydrology*. New York: McGraw-Hill, 1993.
[58] X. Dutoit, B. Schrauwen, J. Van Campenhout, D. Stroobandt, H. Van Brussel, and M. Nuttin, "Pruning and regularization in reservoir computing," *Neurocomputing*, vol. 72, nos. 7–9, pp. 1534–1546, 2009.
[59] S. Yong, L. Yibin, W. Qun, and L. Caihong, "Multi-steps prediction of chaotic time series based on echo state network," in *Proc. IEEE 5th Int. Conf. Bio-Inspired Comput.: Theories Appl.*, Changsha, China, Sep. 2010, pp. 669–672.

Li-Chiu Chang received the B.S., M.S., and Ph.D. degrees in agricultural engineering from National Taiwan University, Taipei, Taiwan, in 1991, 1994, and 2001, respectively.

She is currently an Associate Professor with the Department of Water Resources and Environmental Engineering, Tamkang University, Taipei. She also serves as the Executive Director with the Taiwan Hydro-Informatics Society, Taipei. She has coordinated more than ten government projects related to optimal reservoir operation and an integration platform for flood warning systems in Taiwan. She has published over 40 technical and scientific papers in peer-reviewed journals and co-authored a textbook entitled *Introduction to Artificial Neural Networks*. Her current research interests include water resources management, reservoir operation, artificial intelligence (artificial neural networks, genetic algorithms, fuzzy sets), flood forecasting, and data mining.

Pin-An Chen received the B.S. degree from National Taiwan University, Taipei, Taiwan, in 2009, where he is currently pursuing the Ph.D. degree.

He is currently involved in two research projects granted by the National Science Council, Taiwan: the Development of Novel Static and Dynamic ANNs for Hydro-Environmental Systems, and Sustainable and Integrated Water Resources Management on Watersheds Facing Industrial Pollutions and Urbanization: A Comparative Study between Lot River (France) and Danshuei River (Taiwan). His current research interests include recurrent neural networks, machine learning, data mining, and water quality issues.

Fi-John Chang received the Bachelors and M.S. degrees in agricultural engineering from National Taiwan University, Taipei, Taiwan, and the Ph.D. degree in civil engineering from Purdue University, Lafayette, IN, in 1988.

He is currently a Distinguished Professor with the Department of Bioenvironmental Systems Engineering, National Taiwan University, and the Founding President of the Taiwan Hydro-Informatics Society, Taipei, Taiwan. He is also a Visiting Professor (Scholar) with Zhejiang University, Hangzhou, China, Purdue University, and the University of Illinois at Urbana-Champaign, Urbana. His areas of expertise are hydrology, water resources management, artificial intelligence, hydro-informatics, eco-hydrology, flood forecasting, and water quality estimation modeling. He has published more than 150 papers in peer-reviewed journals and three textbooks on artificial neural networks. He has coordinated more than 60 government projects and has collaborated long term with experts from USA and France on research projects.

Dr. Chang was the recipient of the Outstanding Research Award from the National Science Council, China, in 2010, the Award of Outstanding Contribution to Water Industry from the Ministry of Economic Affairs, China, in 2010, and the three-time recipient of the Outstanding Teaching Award from National Taiwan University.