Hydrol. Earth Syst. Sci., 17, 253–267, 2013
www.hydrol-earth-syst-sci.net/17/253/2013/
doi:10.5194/hess-17-253-2013

Hydrology and Earth System Sciences

Echo state networks as an alternative to traditional artificial neural networks in rainfall–runoff modelling

N. J. de Vos

Prof. Kamerlingh Onneslaan 188, 3112VM Schiedam, The Netherlands

Correspondence to: N. J. de Vos (njdevos@gmail.com)

Received: 1 April 2012 – Published in Hydrol. Earth Syst. Sci. Discuss.: 11 May 2012
Revised: 8 December 2012 – Accepted: 17 December 2012 – Published: 22 January 2013

Abstract. Despite theoretical benefits of recurrent artificial neural networks over their feedforward counterparts, it is still unclear whether the former offer practical advantages as rainfall–runoff models. The main drawback of recurrent networks is the increased complexity of the training procedure due to their architecture. This work uses the recently introduced and conceptually simple echo state networks for streamflow forecasts on twelve river basins in the Eastern United States, and compares them to a variety of traditional feedforward and recurrent approaches. Two modifications on the echo state network models are made that increase the hydrologically relevant information content of their internal state. The results show that the echo state networks outperform feedforward networks and are competitive with state-of-the-art recurrent networks, across a range of performance measures. This, along with their simplicity and ease of training, suggests that they can be considered promising alternatives to traditional artificial neural networks in rainfall–runoff modelling.

1 Introduction

The development of rainfall–runoff (R–R) models that make accurate and reliable forecasts of river streamflow remains among the most important and difficult tasks in hydrology. A plethora of methods exist, such as the popular conceptual models which use simplified descriptions of physical processes. Examples of this approach are the HBV model (Lindström et al., 1997), TOPMODEL (Beven et al., 1995), and the Sacramento soil moisture accounting model (Burnash, 1995). Because of their flexibility and ease of use, data-driven methods based on time series analysis, or, more recently, machine learning methods such as artificial neural networks (ANNs) are increasingly considered as alternatives (e.g. Hsu et al., 1995; Shamseldin, 1997; Campolo et al., 1999; Abrahart and See, 2000; Jain and Srinivasulu, 2004; de Vos and Rientjes, 2008b). Although many investigations of ANNs suggested good performance, their strongly empirical, "black-box" nature limits possible applications, and has raised concerns regarding their reliability and validity as hydrological models (e.g. Cunge, 2003; de Vos and Rientjes, 2005).

Most research on ANNs as R–R models has focused on so-called feedforward ANNs, which perform a static mapping between model input and output. In order to represent the memory of the system in feedforward ANNs, dynamical properties are commonly explicitly modelled by using tapped-delay lines on input variables so that the input space is expanded to a certain time window. Recurrent ANNs, on the other hand, have cyclical connections in the structure of the network that allow an implicit, more parsimonious modelling of dynamical properties. They implement dynamical systems capable of representing and encoding deeply hidden states in which a network's output depends on an arbitrary number of previous inputs, which is why their temporal representation capabilities can be better than those of feedforward ANNs with tapped-delay lines (Saad et al., 1998). Since river basins are dynamical systems, such capabilities seem to give recurrent ANNs a significant advantage over feedforward ANNs in representing a basin's hydrological state. Indeed they have been successfully tested as R–R models by, for example, Hsu et al. (1997), Coulibaly et al. (2000), Chang et al. (2002) and Chiang et al. (2004), but the number of applications using feedforward ANNs dwarfs those with recurrent ANNs. The main reason for this is that recurrency in ANNs causes increased complexity of the training procedure as a result of the cyclical network connections, and subsequent convergence problems for training algorithms (Atiya and Parlos, 2000; Lukoševičius and Jaeger, 2009). As such, it is still not fully clear whether recurrent ANNs, despite theoretical benefits, offer practical advantages over feedforward ANNs with tapped-delay lines in R–R modelling.

Reservoir computing (RC) has recently been introduced as an alternative to traditional recurrent ANNs (Jaeger, 2001). RC commonly involves (1) the generation of a non-adaptable, recurrent ANN whose state maintains a non-linear transformation of its input history (the "reservoir"), and (2) the training of a non-recurrent, usually linear, model that extracts the desired response from the reservoir's state (the "readout"). The training approach of RC methods can therefore be notably simpler and faster than the ones applied to traditional recurrent ANNs. However, it requires that enough process information is contained in the reservoir state for the linear method to extract. RC methods have attracted a lot of interest thanks to their fast training times and good performance compared to traditional methods of system identification, prediction and classification (Hammer et al., 2009; Lukoševičius and Jaeger, 2009). The field is still very young, though, and research on optimal reservoir design and readout methods is ongoing. Very few applications of RC in hydrology have been reported thus far. An early example is Sacchi et al. (2007), who used the most popular RC model, the echo state network (ESN), as one-month-ahead predictor of hydropower plant reservoir inflow, based on historical inflow values. The ESN model was compared to the Self-Organizing Nonlinear Auto-Regressive model with eXogenous input (SONARX) (Barreto and Araújo, 2001), a SONARX variant based on the radial basis function network (SONARX-RBF) (Sacchi et al., 2004) and the adaptive neuro-fuzzy inference system (ANFIS) model (Jang, 1993), with various windows-in-time as input for the latter models. It was concluded that ESN performed significantly better than SONARX, and slightly better than the SONARX-RBF and ANFIS models. Another example of RC methods in the literature is Coulibaly (2010), who used ESNs for forecasting monthly water levels of four North American Great Lakes. It was shown that this model generally outperformed both a standard recurrent ANN and a Bayesian neural network model over a range of forecast lead times. The paper was subsequently discussed and extended by means of a simple linear benchmarking operation by Abrahart et al. (2012b), who showed that the good performance of ESNs can be attributed to their highly dynamical non-linear structure.

This study's main aim is to find out if ESN R–R models can be considered valid alternatives to feedforward and traditional recurrent ANN approaches. Forecasting performance of such models is therefore evaluated on a variety of meso-scale river basins. Experiments are done comparing three different variations of the ESN with two simple benchmark models, two feedforward ANN models, and three recurrent ANN models, each of the latter trained with two recurrent training algorithms. The comparison is made using a variety of performance measures, for one and multiple time steps ahead. Additionally, several reservoir design aspects of ESNs are investigated in order to optimise the hydrologically relevant information contained in the reservoir state, allowing for more accurate and reliable R–R models.

Section 2 briefly reviews feedforward and traditional recurrent ANN models and their training methods, after which a short introduction to RC is given. Section 3 presents the data set and model settings used in this work. The results of these experiments are presented and discussed in Sect. 4, and finally conclusions are drawn in Sect. 5.

2 Artificial neural networks

2.1 Feedforward versus recurrent networks

As shown in Fig. 1a, feedforward ANNs only have forward network connections between the network input(s), the hidden layer(s) of neurons and the output neuron(s). As such, they can be thought to perform a static function mapping between an input **U** and an output **Y** (see Eqs. 1 and 2).

$$y_k = g\left(\sum_{j=1}^{J} x_j w_{jk} + b_k\right) \tag{1}$$

$$x_j = f\left(\sum_{i=1}^{I} u_i w_{ij} + b_j\right) \tag{2}$$

where J and I are the number of hidden neurons and inputs, respectively, y_k is the value of the kth output neuron, x_j is the so-called activation value of the jth hidden neuron, u_i is the ith input value, b is a bias value, w is a connection weight, and f and g are transfer functions. f is almost always non-linear and g either linear or non-linear.

In order to allow system memory to be incorporated into these static ANN models, tapped-delay lines are commonly used, which result in a window of historical values of the variable as input signals (e.g. $U_t, U_{t\text{-}1}, \ldots, U_{t\text{-}s}$). By increasing s, the size of the input vector and therefore the number of connection weights in the ANN are increased, making the model less parsimonious.

Recurrent ANNs represent dynamical systems and are able to model more complex temporal relationships. Figure 1b and c show the two types of traditional recurrent ANN models that are tested in this work, the Elman network (Elman, 1990) and the fully recurrent network (Williams and Zipser, 1989). Both networks have cyclical connections in the structure. The Elman network is a partially recurrent network and has (besides feedforward connections) connections from the hidden neurons that loop back to themselves, fully connected, with a time step delay. The equation for the activation

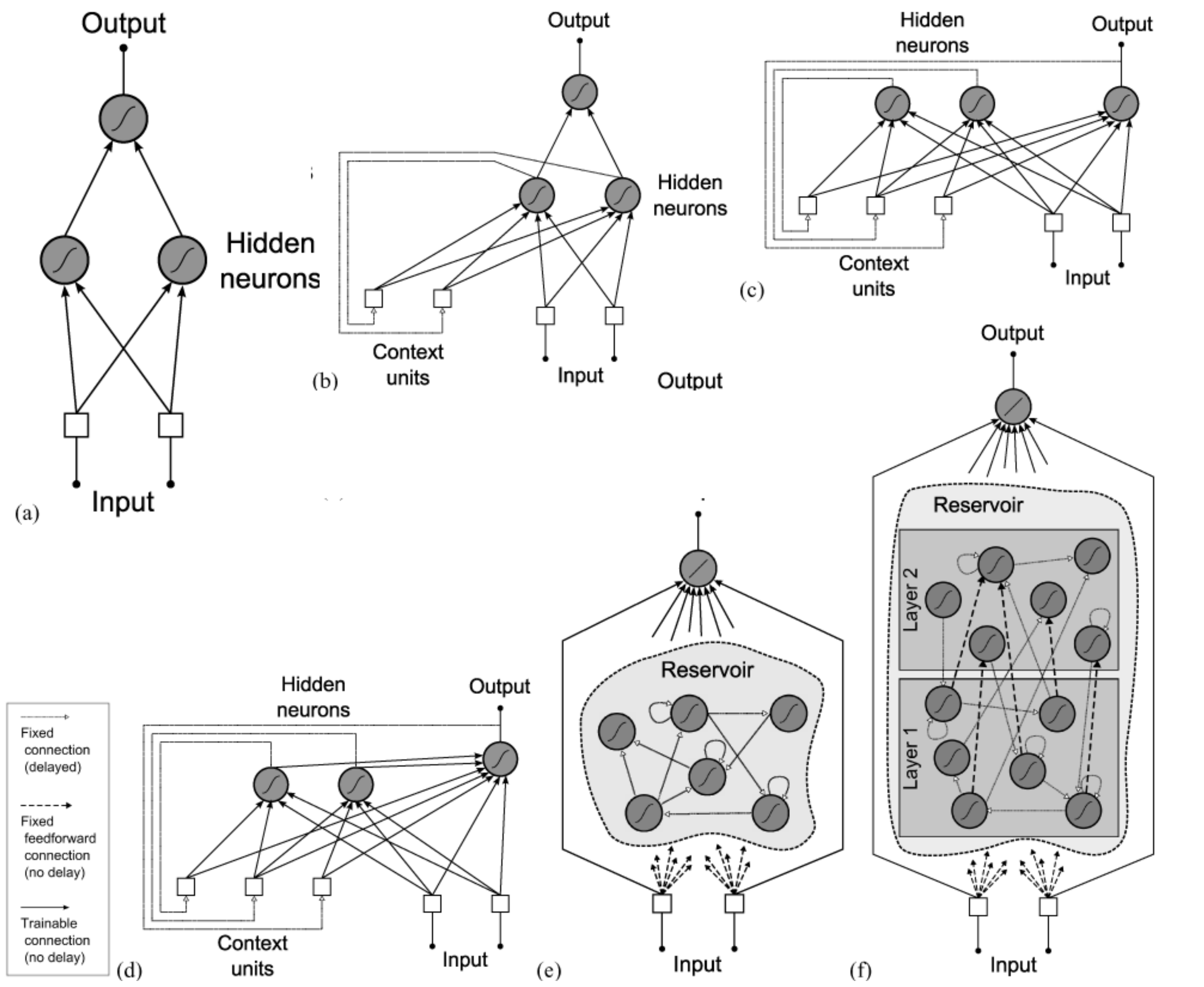

Fig. 1. **(a)** Feedforward ANN. **(b)** Elman recurrent ANN. **(c)** Williams–Zipser fully recurrent ANN. **(d)** Williams–Zipser fully recurrent and fully feedforward ANN. **(e)** Echo State Network. **(f)** Layered Echo State Network. Note: bias signals are omitted for clarity.

function thus becomes

$$x_j^t = f\left(\sum_{i=1}^{I} u_i^t w_{ij} + \sum_{k=1}^{K} x_k^{t\text{-}1} w_{kj} + b_j\right). \tag{3}$$

In the Williams–Zipser fully recurrent ANN, the input connects directly to all hidden and output neurons, and the total of hidden and output neurons is fully interconnected with a time step delay.

2.2 Training methods

A common approach to ANN training in function approximation applications such as R–R modelling is to use supervised training. Sample input and output data are presented to the network, after which optimisation algorithms attempt to minimise the network output error by adjusting the network weight matrix **W**. The most popular training method for feedforward ANNs is the standard backpropagation (BP) algorithm (Rumelhart and McLelland, 1986), which uses a first-order gradient-descent method to find optimal weight values. The objective function $E(\mathbf{W})$ is calculated after which the BP algorithm applies an update to the weights in the network:

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}}, \tag{4}$$

where η is the learning rate of the BP algorithm. The weight updating corresponds to following the gradient along the error surface $E(\mathbf{W})$ in search for a minimum. Weight updates can be performed every time a single training pattern has been presented (i.e. online mode), or based on the mean error over all training data (i.e. batch mode).

More sophisticated alternatives to the standard BP algorithm, such as the conjugate gradient (CG) algorithm and the second-order-gradient-based Levenberg–Marquardt

(LM) algorithm, have been found to commonly outperform in terms of accuracy and convergence speed (e.g. Møller, 1993; Hagan and Menhaj, 1994; de Vos and Rientjes, 2005). In the LM algorithm, weight updates are performed according to

$$\Delta \boldsymbol{w} = -[\mathbf{H}+\mu\mathbf{I}]^{-1}\mathbf{J}^T\boldsymbol{e} \tag{5}$$

where μ is a (variable) learning rate, $\mathbf{J}$ the Jacobian matrix that contains first derivatives of the network errors with respect to the weights, $\boldsymbol{e}$ a vector of network errors, and $\mathbf{H}$ an approximation of the Hessian matrix, $\mathbf{H}=\mathbf{J}^T\mathbf{J}$. More information can be found in (Hagan and Menhaj, 1994).

Recurrent ANN training has also traditionally relied on gradient-based methods. The most commonly used algorithm, backpropagation through time (BPTT), is an extension of the standard BP method (see Werbos, 1990). BPTT also uses a first-order-gradient approach to weight updates. The recurrent connections inside the network are dealt with by unfolding time iterations of the network into layers, creating an equivalent feedforward network (Atiya and Parlos, 2000). Like BP, training can be performed in batch mode or online mode. Using the latter, the network unfolding is limited to a chosen truncation depth to keep the method computationally feasible (Haykin, 1999).

The extended Kalman filter (EKF) is a well-known method for non-linear state estimation of dynamical systems that has been successfully used for recurrent ANN training (Puskorius and Feldkamp, 1994; Sum et al., 1998). Following the notation in Haykin (2001), the dynamics of the recurrent network are modelled as

$$\begin{aligned}\mathbf{W}_{t+1} &= \mathbf{W}_t + \boldsymbol{\omega}_t\\ \mathbf{Y}_t &= \boldsymbol{h}_t(\mathbf{W}_t,\mathbf{U}_t,\boldsymbol{\upsilon}_{t\text{-}1}) + \boldsymbol{\upsilon}_t,\end{aligned} \tag{6}$$

where $\mathbf{W}$ are the network weights, $\mathbf{U}$ the input, $\boldsymbol{\omega}$ and $\boldsymbol{\upsilon}$ are Gaussian uncorrelated noises representing process and measurement noise, respectively, and $\mathbf{Y}$ is the output, which is based on a time-dependent function $\boldsymbol{h}_t$. The task of the EKF is to estimate optimal weights, given a series of observed outputs. This is done at each time step through so-called measurement updates of the EKF procedure:

$$\begin{aligned}\mathbf{K}_t &= \mathbf{P}_t\mathbf{H}_t\left(\frac{1}{\eta}\mathbf{I}+\mathbf{H}_t^T\mathbf{P}_t\mathbf{H}_t+\mathbf{R}_t\right)^{-1}\\ \mathbf{W}_{t+1} &= \mathbf{W}_t+\mathbf{K}_t\xi_t\\ \mathbf{P}_{t+1} &= \mathbf{P}_t-\mathbf{K}_t\mathbf{H}_t^T\mathbf{P}_t+\mathbf{Q}_t,\end{aligned} \tag{7}$$

where $\mathbf{K}$ is called the Kalman gain, $\mathbf{P}$ is the error covariance matrix of the weights, $\mathbf{Q}$ is the covariance matrix of the noise $\boldsymbol{\omega}$, $\mathbf{R}$ is the covariance matrix of the noise $\boldsymbol{\upsilon}$, η is a learning rate parameter, ξ is the difference between observed output and output calculated from the previous weight estimate, and $\mathbf{H}$ contains the derivatives of the network output with respect to the weights. The latter can be calculated using the truncated BPTT procedure mentioned above.

Traditional RNN training using BPTT, and even the sophisticated EKF, suffers from several shortcomings related to the combination of model complexity and gradient-based optimisation (after Lukoševičius and Jaeger, 2009):

- Gradual weight updates during the training procedure may drive recurrent networks through bifurcations where gradient information becomes useless (Doya, 1992).
- Weight updates can be computationally expensive and many updates may be necessary.
- Relationships over long-range memory are hard to learn, because the necessary gradient information exponentially dissolves over time (Bengio et al., 1994).
- Training algorithms require skill and experience to apply, since their complexity requires a number of global control settings that are not easily optimised.

2.3 Reservoir computing

2.3.1 History and developments

Reservoir computing refers to a group of recurrent ANN methods that share certain specific aspects of network design and training that are notably different from traditional methods. Its primary examples are the independently proposed but recently unified (see Verstraeten et al., 2007) liquid state machines (Maass et al., 2002), ESNs (Jaeger, 2001; Jaeger and Haas, 2004) and Backpropagation-Decorrelation learning rule (Steil, 2004). In this work ESNs are used, which are the simplest and most common approach to RC.

Recently, RC methods have attracted a lot of research interest because they overcome several of the training problems of traditional recurrent ANN methods by separating the simulation of model dynamics and the training of the network. As a consequence, they offer excellent modelling accuracy and fast training times (see review by Lukoševičius and Jaeger, 2009). Additionally, RC methods are easily extensible for additional outputs. However, significant research challenges remain, concerning richness of reservoir dynamics, optimal readouts and (both general and task-specific) model design guidelines.

2.3.2 Reservoir and readout workings

RC involves (1) the generation of the so-called reservoir, a non-adaptable recurrent ANN whose state maintains a non-linear transformation of its input history, and (2) the training of a non-recurrent, usually linear, readout that extracts the desired response from the reservoir's state (see Fig. 1e for an example RC network). The idea behind this is that the reservoir contains a representation of current and historical system dynamics that is rich enough to enable the readout to learn the functional dependence between system input and output.

This approach can be thought of as non-linearly and temporally expanding the input into a high-dimensional feature vector and then utilizing those features using linear methods (Lukoševičius and Jaeger, 2009), in the same vein as methods that rely on kernel expansion (e.g. Support Vector Machines).

A reservoir functions like a regular recurrent ANN, without the requirement that the neurons are arranged in layers. Its architecture is not strictly defined, and can be custom-made. It is not formally required, but generally intended that the internal connections induce recurrent pathways between neurons (Jaeger, 2001). Often a large, sparsely connected reservoir with random weights is used, based on the intuition that such a network should be able to maintain a rich dynamical state. However, what exactly constitutes rich dynamics in the network's activation values is generally not well-defined and depends on the modelling task at hand. Moreover, because of the strong coupling between activation values, a reservoir often lacks the ability to represent multiple time scales simultaneously. To overcome such problems, various variations on reservoir design have been proposed (see review by Lukoševičius and Jaeger, 2009), but no sharp guidelines for optimal reservoir design exist as of yet.

A condition called the 'echo state property' is imposed on the reservoir of an ESN, which states that the effect of previous inputs and states on future states should vanish gradually as time passes, and not persist or even get amplified (see Jaeger, 2001; Buehner and Young, 2006, for details). By scaling the reservoir weight matrix **W** according to its spectral radius (i.e. largest absolute eigenvalue) $\rho(\mathbf{W}) < 1$, this condition is generally satisfied. The value of the spectral radius is intimately connected to the intrinsic time scale of the dynamics of the reservoir state (Jaeger, 2002). Optimal values for $\rho(\mathbf{W})$ thus depend on the degree of non-linearity and memory that a model requires.

Generally, linear readouts are used because they have the advantage that they can be trained by well-studied and fast linear regression methods. A popular example is ridge regression (a.k.a. Tikhonov regularisation), which is based on the idea of adding an additional cost term to the least squares optimisation so that the norm of the weights is kept small. It has been proven successful in improving robustness and generalisation of RC models (Wyffels et al., 2008). In ridge regression, weight values are determined according to

$$\mathbf{W} = \mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \alpha^2\mathbf{I})^{-1}, \tag{8}$$

where **Y** is the target output, **X** the reservoir state, and α a regularisation parameter.

2.3.3 On instantaneous non-linear capacity of a reservoir

Lukoševičius (2007) and Lukoševičius and Jaeger (2009) have pointed out that the traditional ESN architecture can suffer from an inability to produce a both non-linear and instantaneous mapping from input to output. Feedforward and partially recurrent ANNs such as the Elman network do have this ability thanks to feedforward connections through one or more hidden layers. Traditional ESN output, however, is a linear combination of model input and reservoir state. Although a reservoir allows for non-linear transformations of the input, this signal is mixed with previous values of the reservoir internal state.

A possible solution to this problem was introduced by Lukoševičius (2007) in the form of layered echo state networks (LESNs). An LESN's reservoir is divided into L layers with roughly the same amount of neurons, and each time step it is updated layer by layer. Figure 1f shows the organisation of layers and connections inside a LESN. At each time step, the activation values of the neurons in layer 1 are calculated first, based on the input signals and the delayed recurrent signals from all neurons (i.e. exactly like in a regular ESN). Subsequent layers' activation values are calculated from the input, all previous layers' present activation values and all recurrent signals. In this way, LESNs attempts to combine the benefits of feedforward and recurrent connections by allowing the input signals to propagate forward through multiple layers of neurons during a single time step. Increasing L enables more complex instantaneous transformation at the cost of reduced memory capacity.

The Williams–Zipser fully recurrent ANN to a certain degree suffers from the same problem because of the lack of non-delayed connections between input and output layer through a hidden layer. This work therefore introduces a variation on this network which does have feedforward connections between the hidden layer and the output layer (the WZFF, see Table 2 and Fig. 1d).

2.3.4 On reservoir dynamics

Ever since the introduction of ESNs, leaky-integrator neurons have been suggested for allowing a reservoir to represent slower dynamics (Jaeger, 2001). Leaky-integrator neurons apply a low-pass filter in the form of an exponential moving average to its activation values, as shown in Eq. (9) (cf. Eq. 3). The coefficient $c \in [0,1]$ is the decay rate, for which a value of 1 results in a regular neuron.

$$x_j^t = (1\text{-}c)x_j^{t-1} + f\left(\sum_{i=1}^{I} u_i^t w_{ij} + \sum_{k=1}^{K} x_k^{t\text{-}1} w_{kj} + b_j\right) \tag{9}$$

Table 1. Overview of MOPEX river basins and their characteristics.

ID	River	Latitude	Longitude	Area (km^2)	Mean elev. (m)	Soil	Vegetation	Mean annual precip. (mm)	Mean annual pot. evap. (mm)	Mean annual Q (mm)	Skew-ness Q
B1	S. Branch Potomac	39.4469	−78.6544	3810	171	Loam	Dec. broad leaf	1042	761	385	16.0
B2	Monocacy	39.3880	−77.3800	2116	71	Silt loam	Dec. broad leaf	1041	896	420	11.0
B3	Rappahannock	38.3222	−77.5181	4134	17	Clay loam	Mixed forest	1030	920	377	8.0
B4	Tygart Valley	39.1500	−80.0400	2372	390	Loam	Dec. broad leaf	1166	711	745	4.2
B5	Bluestone	37.5439	−81.0106	1020	465	Silt clay loam/loam	Dec. broad leaf	1018	741	421	5.3
B6	East Fork White	39.2000	−85.9256	4421	184	Silt loam/clay loam	Cropland	1015	855	376	5.2
B7	French Broad	35.6092	−82.5786	2448	594	Loam	Mixed forest	1383	819	796	4.2
B8	English	41.4664	−91.7156	1484	193	Clay loam	Cropland	893	994	272	6.9
B9	Spring	37.2456	−94.5661	3015	254	Silt loam/clay loam	Dec. broad leaf	1076	1094	302	12.5
B10	Amite	30.4639	−90.9903	3315	0	Silt loam	Ever. needleleaf	1564	1073	584	7.4
B11	Guadelupe	29.8606	−98.3828	3406	289	Clay	Crop/nat. veg.	765	1528	116	25.6
B12	San Marcos	29.6650	−97.6497	2170	98	Clay	Crop/nat. veg.	827	1449	179	35.0

3 Experimental setup

3.1 Data

Following the recent call for the use of benchmark data sets in ANN research by Abrahart et al. (2012a), the model simulations in this work are done on the well-known Model Parameter Estimation Experiment (MOPEX) data set as presented in Duan et al. (2006). This data set includes daily precipitation, potential evaporation, and discharge data for twelve river basins in the Eastern United States. Table 1 shows geographical, hydrometeorological and land surface characteristics of these basins. The skewness of the discharge data is shown in the last column. High skewness values indicate occurrence of extreme high flow events.

The data were split up into training (1979–1990, with the first year for model spin-up), cross-validation (1991–1998) and test (1960–1979) periods.

3.2 Model input and output

The data used as inputs to the ANN models are time series of daily precipitation (P), potential evaporation (E), discharge (Q), and the 20-day simple moving average of the precipitation time series (P_{ma}). The latter serves as a crude indicator of the wetness in the basin.

As explained in Sect. 2, the Elman, Williams–Zipser and ESN networks use only the latest values of the P, E, Q, and P_{ma} data as input. Two feedforward models were constructed, one of which also uses only these latest values of the 4 variables, and the other used tapped-delay lines. In order to determine the optimal input windows for these tapped-delay lines, linear correlation and average mutual information between the input and output time series were calculated. These results suggested that the information content with respect to Q at time $t+1$ for all catchments was highest in the last three time steps for both P and Q. Moreover, because of the relatively low time frequency of the data compared to the catchment dynamics, there is high linear (auto-)correlation at t_0 and t-1 and a sharp drop after that. For E and P_{ma}, correlations were flatter and only time step t_0 was chosen. This results in a total of 8 inputs, as shown in Table 2.

All input data were standardised to have a mean of 0 and a standard deviation of 1, and the output data were normalised to a range of −0.9 to 0.9.

3.3 Training settings

The feedforward ANNs were trained in batch mode by the LM algorithm. Standard backpropagation was tested but not presented here because it performed very poorly compared to the LM algorithm, most likely due to the first-order gradient algorithm getting stuck in local optima. The Elman and Williams–Zipser ANNs were trained online by both truncated BPTT and EKF (with truncated BPTT for determining the necessary gradients). BPTT used a window size of 10 and a learning rate of 0.005. For EKF, a learning rate η of 1 was used, and the covariance matrix $\mathbf{P}$ was initialised with values of 1000 on the diagonal and 0 elsewhere. The initial matrices for $\mathbf{Q}$ and $\mathbf{R}$ used values of 10^{-4} and 200, respectively. Before training, feedforward, Elman and Williams–Zipser ANN weights were drawn randomly from a uniform distribution where $w \in [-0.25, 0.25]$. The RC network readouts were trained in batch mode by ridge regression. Unless noted otherwise, all presented summary statistics are based on 20 independent training runs, to properly reflect the effects of randomness in initialisation and/or training of the model weights.

A potential pitfall of ANN training is overfitting, which means that the network has learned the intricacies of the training data, including noise, and thereby has lost its ability to generalise beyond the specifics of this data. In order to improve their generalisation capability, training of the feedforward, Elman and Williams–Zipser models used the often-applied early-stopping approach, in which weight adaptation

Table 2. Overview of the rainfall–runoff models used.

Model	Architecture	Inputs	Network structure$^+$	No. of weights	Training method
PM	Persistence model	Q_t	–	–	–
LIN	Multiple linear regression model	$P_{\mathrm{ma},t}$, $P_{t\text{-}2}$, $P_{t\text{-}1}$, P_t, E_t, $Q_{t\text{-}2}$, $Q_{t\text{-}1}$, Q_t	–	9	Ridge regression
FF	Feedforward ANN	$P_{\mathrm{ma},t}$, P_t, E_t, Q_t	4–2–1	13	LM
$\mathrm{FF}_{\mathrm{TDL}}$	Feedforward ANN w/ tapped-delay lines	$P_{\mathrm{ma},t}$, $P_{t\text{-}2}$, $P_{t\text{-}1}$, P_t, E_t, $Q_{t\text{-}2}$, $Q_{t\text{-}1}$, Q_t	8–3–1	31	LM
$\mathrm{EL}_{\mathrm{BPTT}}$	Elman recurrent ANN	$P_{\mathrm{ma},t}$, P_t, E_t, Q_t	4–4–1	41	BPTT
$\mathrm{EL}_{\mathrm{EKF}}$	Elman recurrent ANN	$P_{\mathrm{ma},t}$, P_t, E_t, Q_t	4–4–1	41	EKF
$\mathrm{WZ}_{\mathrm{BPTT}}$	Williams–Zipser fully recurrent ANN	$P_{\mathrm{ma},t}$, P_t, E_t, Q_t	4–4–1	50	BPTT
$\mathrm{WZ}_{\mathrm{EKF}}$	Williams–Zipser fully recurrent ANN	$P_{\mathrm{ma},t}$, P_t, E_t, Q_t	4–4–1	50	EKF
$\mathrm{WZFF}_{\mathrm{BPTT}}$	Williams–Zipser fully recurrent and fully feedforward ANN	$P_{\mathrm{ma},t}$, P_t, E_t, Q_t	4–4–1	54	BPTT
$\mathrm{WZFF}_{\mathrm{EKF}}$	Williams–Zipser fully recurrent and fully feedforward ANN	$P_{\mathrm{ma},t}$, P_t, E_t, Q_t	4–4–1	54	EKF
ESN	Echo State Network	$P_{\mathrm{ma},t}$, P_t, E_t, Q_t	4–200–1	≈ 9000 untrained and 205 trained*	Ridge regression
LESN	Layered Echo State Network with 2 layers	$P_{\mathrm{ma},t}$, P_t, E_t, Q_t	4–200–1	≈ 9000 untrained and 205 trained*	Ridge regression
LESN-L	Layered Echo State Network with 2 layers of leaky-integrator neurons	$P_{\mathrm{ma},t}$, P_t, E_t, Q_t	4–200–1	≈ 9000 untrained and 205 trained*	Ridge regression

$^+$ Network structure is expressed as: number of input units – number of units in hidden layer or reservoir – number of neurons in output layer.
* The untrained weights are the randomly generated input-to-reservoir and internal reservoir weights. The trained weights are the readout weights.

is stopped when the error on the cross-validation data starts to increase significantly. The linear regression methods of the RC models exclude the use of early-stopping, so the cross-validation data was not used in their training. Instead, these models relied on regularisation by the ridge regression method for good generalisation ability. Using trial-and-error testing, a good value of the regularisation parameter α (see Eq. 8) was found to be 0.1.

The objective function used for training all ANN models is the mean squared error (MSE) (Eq. 10).

$$\mathrm{MSE} = \frac{1}{T}\sum_{t=1}^{T}(\hat{Q}_t - Q_t)^2 \tag{10}$$

where $\hat{Q}$ is the estimated and Q the observed discharge value.

3.4 Performance evaluation

Model performance is evaluated using the well-known Nash–Sutcliffe coefficient of efficiency of both regular flow values (CE, Eq. 11) and log-transformed flow values ($\mathrm{CE}_{\log}$, Eq. 12), and the Mean Squared Derivative Error (MSDE, Eq. 13). The CE is a scaled variation of the MSE and stresses fit on peak flows, whereas the $\mathrm{CE}_{\log}$ focuses on low flows. The MSDE penalises hydrograph shape errors, especially timing errors and noise, thereby exploiting information that is largely ignored by traditional hydrological model performance measures (de Vos and Rientjes, 2008b). It ignores absolute differences between the simulated and observed hydrographs, but instead expresses differences in their first derivatives. The MSDE can be interpreted as a more accurate variation of the well-known statistic that counts the number of sign changes in the sequence of residuals, as used by the National Weather Service (Brazil, 1988).

$$\mathrm{CE} = 1 - \frac{\sum_{t=1}^{T}(\hat{Q}_t - Q_t)^2}{\sum_{t=1}^{T}(Q_t - \bar{Q}_t)^2} \tag{11}$$

$$\mathrm{CE}_{\log} = 1 - \frac{\sum_{t=1}^{T}(\ln(\hat{Q}_t) - \ln(Q_t))^2}{\sum_{t=1}^{T}(\ln(Q_t) - \ln(\bar{Q}_t))^2} \tag{12}$$

$$\mathrm{MSDE} = \frac{1}{T}\sum_{t=1}^{T}((\hat{Q}_t - \hat{Q}_{t\text{-}1}) - (Q_t - Q_{t-1}))^2, \tag{13}$$

where $\bar{Q}$ is the mean observed discharge value.

As a baseline reference for comparing model performance, a persistence model and a multiple linear regression model were used. The persistence model merely copies the last known value of variable Q as its prediction, creating a lagged copy of the original time series. Comparison with this simple model allows for a more strict and appropriate evaluation of ANN model performance because ANN R–R models have been shown to be prone to the problem of merely using the last known discharge value in their prediction (Anctil et al., 2004; de Vos and Rientjes, 2005; Abrahart et al., 2007; de Vos and Rientjes, 2008a). The multiple linear regression model is based on the same 8 inputs used for the feedforward model with tapped-delay lines (see Table 2) and is also calibrated using ridge regression.

3.5 Artificial neural network design

Through a trial-and-error approach, 2 and 3 neurons in a single hidden layer were found to be optimal for the feedforward ANNs without and with tapped-delay lines, respectively. Also, bias signals are used in the hidden layer and output layer, bringing the total number of weights to be trained to 13 and 31. The optimal number of hidden neurons for the Elman network and the regular and feedforward variations of the Williams–Zipser ANN was determined to be 4, resulting in 41, 50 and 54 weights, respectively (including bias signals). All ANNs in this study used the hyperbolic tangent transfer function in both the hidden and output neurons. Although linear functions are often successfully used in output neurons (e.g. Yonaba et al., 2010), the hyperbolic tangent function was used for the sake of a fair comparison with the Williams–Zipser networks, which use them in all their neurons.

Initial tests suggested that the RC networks performed best if the input signals, a bias signal with value 1, and the reservoir were all fully and directly connected to the readout (as shown in the example in Fig. 1e). The input and bias were also fully connected to the reservoir. No feedback connections from output to reservoir were used, since this seemed to deteriorate performance. The weights of the input-to-reservoir connections determine how strongly a reservoir is excited by input, and thereby the degree of non-linearity of its response. Here these weights were drawn randomly from a uniform distribution where $w \in [-0.1, 0.1]$. All other connection weights were drawn from a normal distribution. The common practice of using a sparsely and randomly connected reservoir is followed, by randomly allowing approximately 20 % of all connections to be active.

The size of the reservoir determines to a large degree the capacity of a network to learn complex dynamics with reasonable accuracy. Additionally, the spectral radius of the reservoir weights controls reservoir dynamics and therefore is an important setting for a RC model (Jaeger, 2002). Figures 2 and 3 show training and cross-validation performance, respectively, over a range of values for both parameters. The results show that large reservoirs, especially in combination with large spectral radii, fit the training data best, but generally at the expense of the cross-validation performance. This is an indication that RC models can be overfitted to the training data, despite regularisation by the ridge regression procedure. A moderate reservoir size of 200 and spectral radius of 0.6 were chosen for further simulations to avoid such problems. Although ignored here in order to allow fair comparison with other ANNs, there are sometimes significant differences between the optimal parameter values for each of the basins, indicating different system dynamics, complexities, and degrees of overfitting.

The LESN model that allows for instantaneous non-linear transformations, as discussed in Sect. 2.3.3, was trained with a range of settings for the number of layers L and evaluated on the cross-validation performance, which showed that the optimal number of layers was 2. To improve reservoir dynamics, Eq. (9) was tested on the recurrent connections of the LESNs (but not the feedforward connections, as not to reduce their instantaneous non-linear capabilities), resulting in the LESN-L model. Initial tests showed that good values for the coefficient c vary between basins due to differences in hydrometeorological drivers and basin characteristics. Therefore, in order to encompass a range of possible system dynamics, the 100 neurons of each of the layers of the LESN were assigned coefficients that were evenly distributed between 0.01 and 1.

4 Results and discussion

Figures 4 and 5 show the CE for the training and test data, respectively, for all models mentioned in Table 2. The overall performance of all ANNs is quite good, judging from the comparison with the persistence model (PM) and multiple linear regression (LIN) benchmark models and the high absolute range of CE values. The performance also compares well to the results presented in Duan et al. (2006), who use the same data set for calibrating 8 well-known conceptual models over the entire available history. Figure 12 of that paper shows that the models' mean Nash–Sutcliffe values over the 12 catchments range between 0.38 and 0.71, and 4 of these models even have a mean below 0.55.

Moreover, results are largely consistent between the training and test data, which means that the models having kept their ability to generalise and are not significantly overfitted. Some river basins (especially B11 and B12) seem to be difficult to model, judging from the low performance of the PM and LIN models. This may be because these catchments are relatively dry and suffer more from flash floods (see Table 1). Additionally, extrapolation issues arise, specifically in basins B2, B3 and B11. These all have one extreme event occurring in the test data that falls significantly outside the range

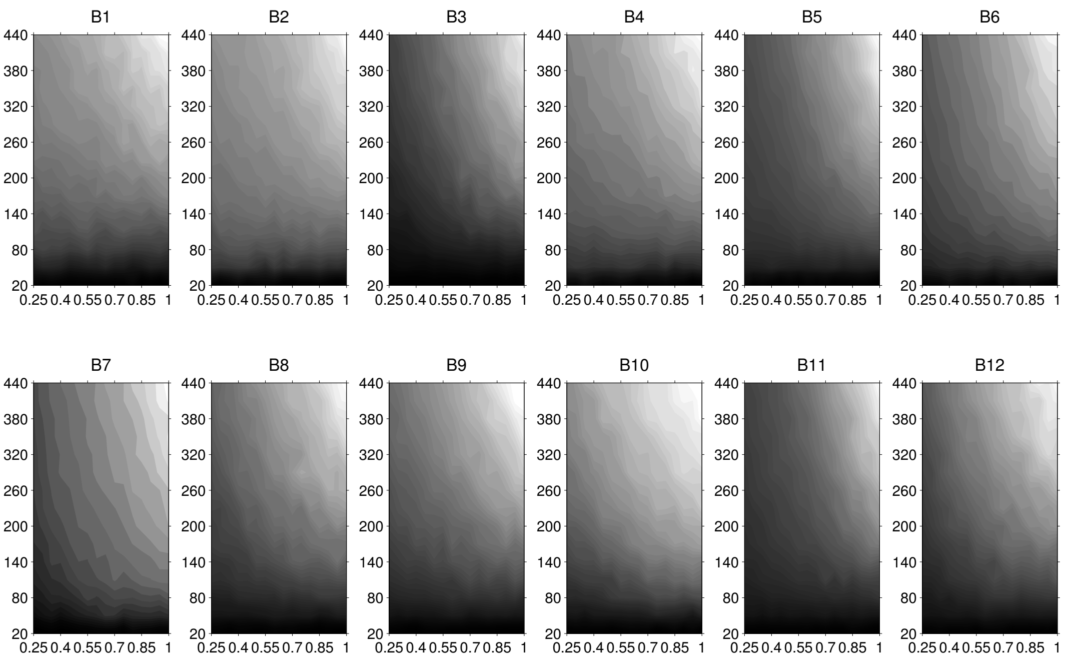

Fig. 2. ESN training performance (averaged over 10 runs) over a range of values for the reservoir size (y-axis) and spectral radius (x-axis). Lighter shades represent a low MSE, darker shades a high MSE.

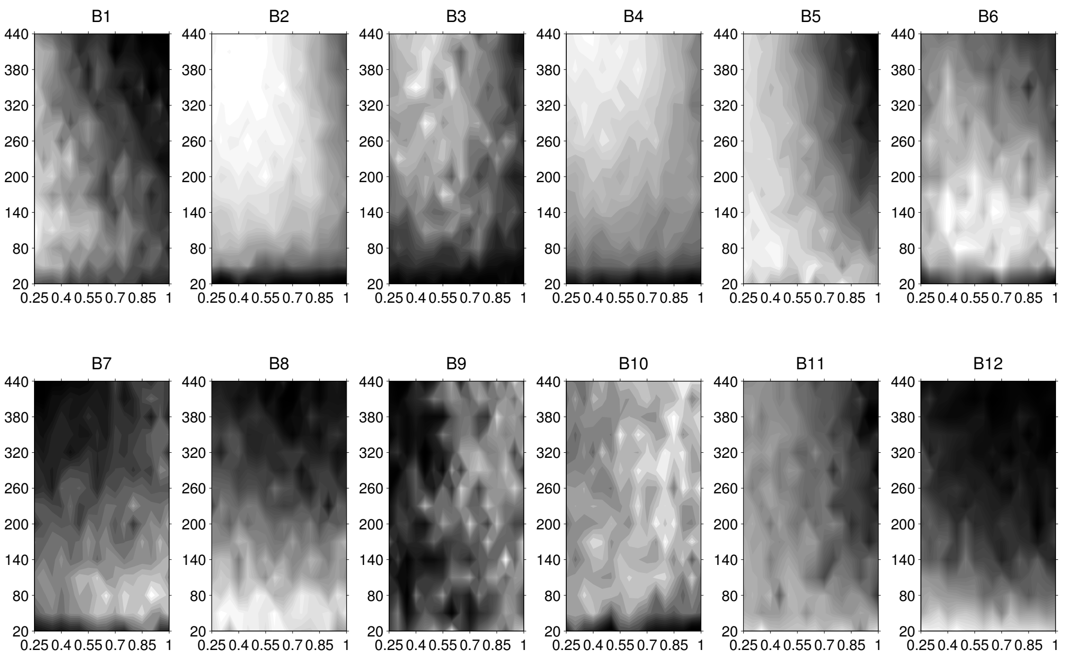

Fig. 3. ESN cross-validation performance (averaged over 10 runs) over a range of values for the reservoir size (y-axis) and spectral radius (x-axis). Lighter shades represent a low MSE, darker shades a high MSE.

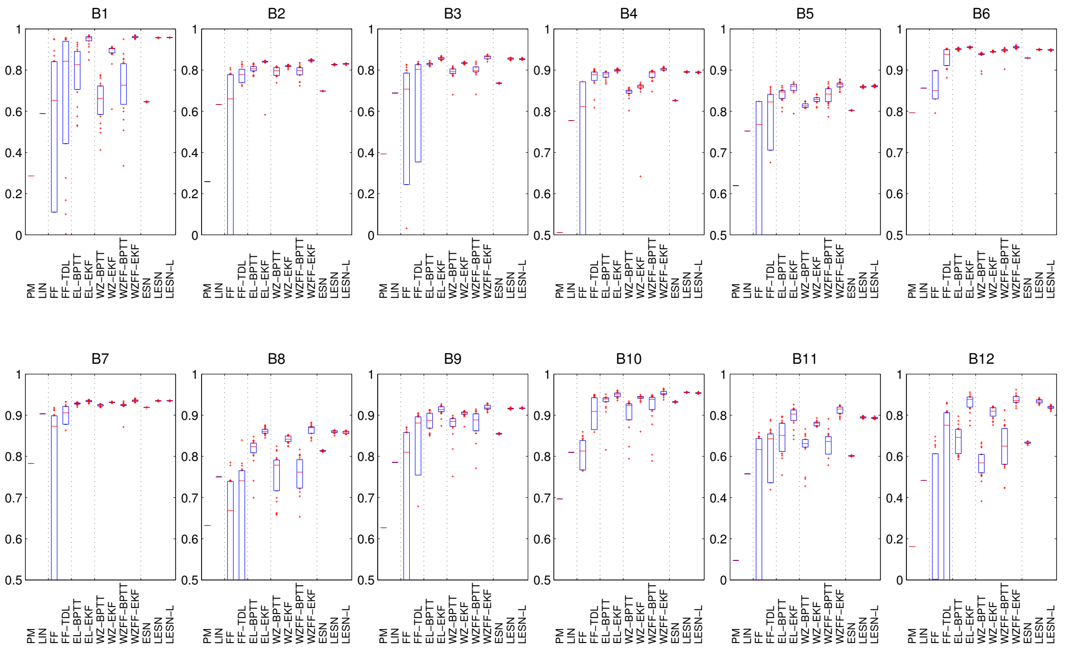

Fig. 4. Training performance in terms of CE. The box plot expresses statistics of over 20 models runs. The central mark is the median, the edges of the box are the first and third quartiles, and points outside that range are plotted individually as dots. Note that not all subfigures have the same scale.

for which the models have been trained, and that attributes significantly to a lower CE.

The feedforward ANNs both generally perform reasonably well and did not require a lot of computational effort. Their non-linearity proves to have added value over the LIN model. However, their spread in performance is often large, which is likely due to the LM algorithm getting stuck in local optima. The tapped-delay lines of the FF_{TDL} network help performance on several basins (especially B6 and B10), while performance on others stays largely the same.

All (partially) recurrent models that have non-linear feedforward connections between input and output (EL, WZFF, and LESN) clearly produce superior results over the models that do not (WZ and ESN). In the meso-scale catchments and for a forecast lead time of one day, the ANNs get most of their information for $Q(t+1)$ from $Q(t)$ and $P(t)$, as underlined by the good performance of the FF model. The latter group's relatively poor performance therefore seems to be attributable to the lack of a sufficiently non-linear, instantaneous mapping of these variables (see discussion in Sect. 2.3.3).

The BPTT training algorithm shows poor performance on most basins, both for the Elman and Williams–Zipser ANNs. This algorithm was also found to be the slowest of all algorithms tested. These results confirm the drawbacks of gradient methods for recurrent ANNs, as discussed in Sect. 2.2. EKF, on the other hand, proves to be a powerful training method that generally allows the Elman and Williams–Zipser networks to utilise their recurrent architectures and outperform the feedforward networks.

EL_{EKF} and $WZFF_{EKF}$ generally outperform FF_{TDL}, suggesting that recurrent ANNs are better models for dealing with system dynamics than feedforward ANNs with tapped-delay lines. However, the EKF algorithm takes significantly more training time and it needs a lot of settings to be tweaked.

Despite randomness in their reservoir construction, the RC models' performance are more consistent than any other model group. Computational effort was higher than the feedforward ANNs, due to the simulation of the reservoir, but less than the other recurrent models, thanks to the fast ridge regression training procedure.

Figures 6 and 7 respectively show the CE_{log} and MSDE performance for the test data, allowing for a more rigorous evaluation of the various models. Both fits on low flow (as judged from the CE_{log} values) and hydrograph shape (as judged from the MSDE values) largely confirm the CE results, indicating reliability of its findings. Again, EL_{EKF}, $WZFF_{EKF}$ and LESN seem to be the best models, and differences between them are small.

For some models or algorithms there seems to be a trade-off between performance measures. The most clear example of this is the performance of BPTT which sometimes scores very well on CE_{log}. Apparently, the BPTT algorithms does find its way to local optima that simulate low flows well, but is unable to fine-tune to also fit the more challenging peak flows.

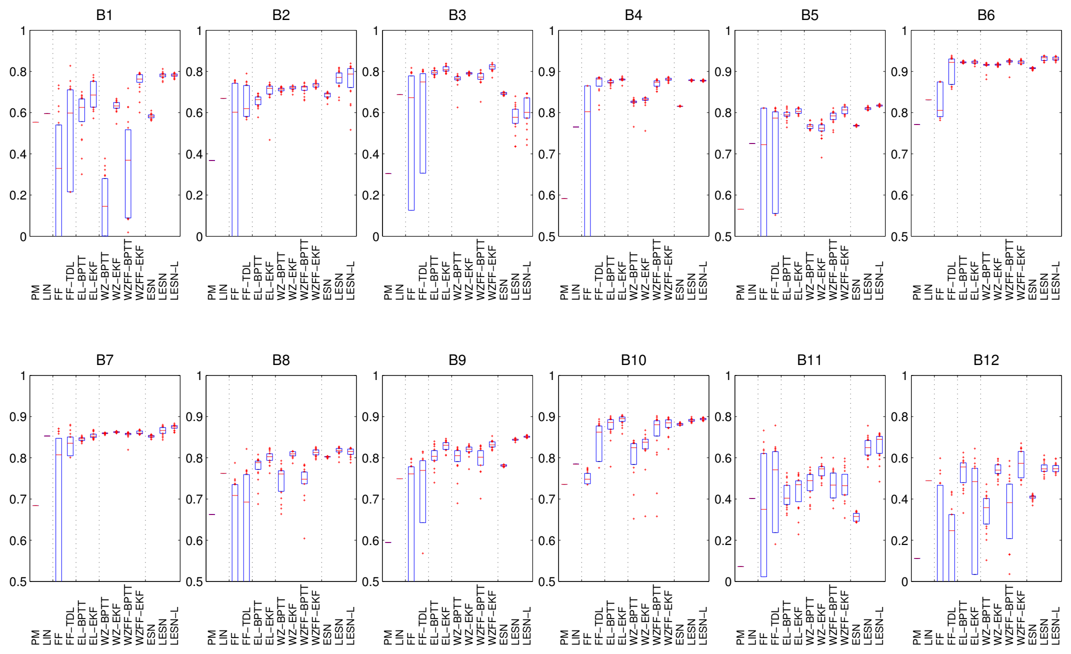

Fig. 5. Test performance in terms of CE. The box plot expresses statistics of over 20 models runs. The central mark is the median, the edges of the box are the first and third quartiles, and points outside that range are plotted individually as dots. Note that not all subfigures have the same scale.

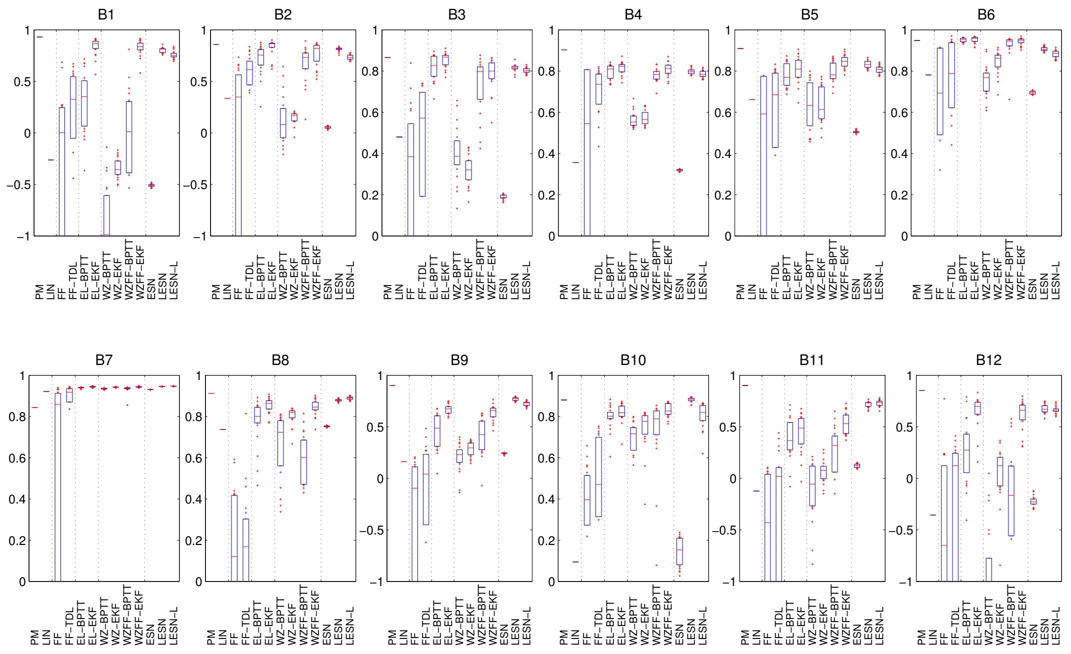

Fig. 6. Test performance in terms of CE_{log}. The box plot expresses statistics of over 20 models runs. The central mark is the median, the edges of the box are the first and third quartiles, and points outside that range are plotted individually as dots. Note that not all subfigures have the same scale.

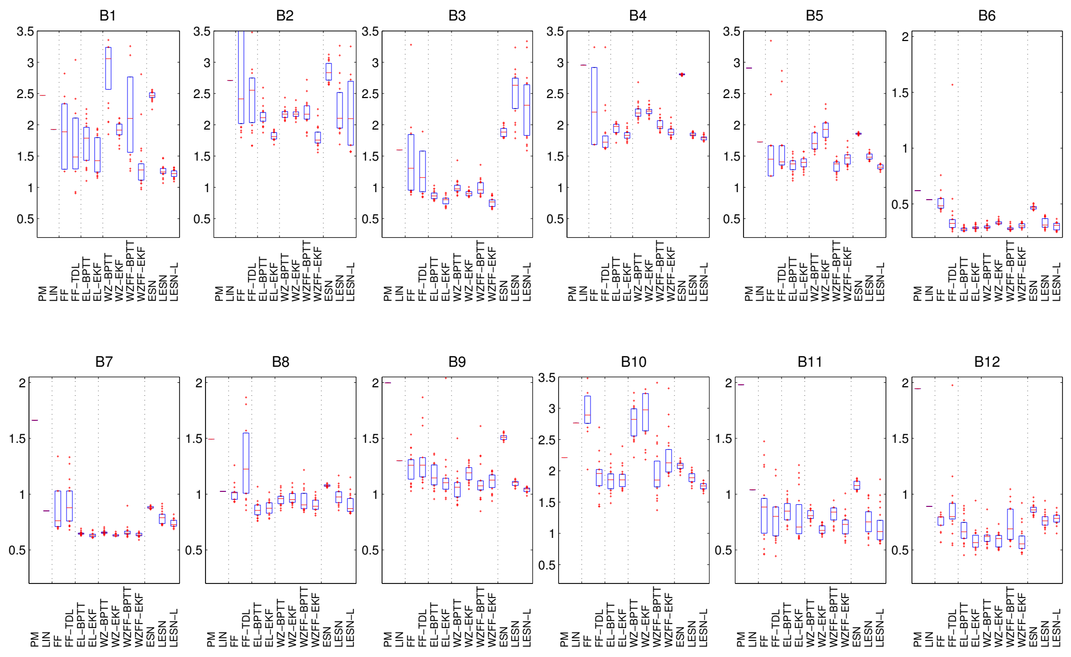

Fig. 7. Test performance in terms of MSDE. The box plot expresses statistics of over 20 models runs. The central mark is the median, the edges of the box are the first and third quartiles, and points outside that range are plotted individually as dots. Note that not all subfigures have the same scale.

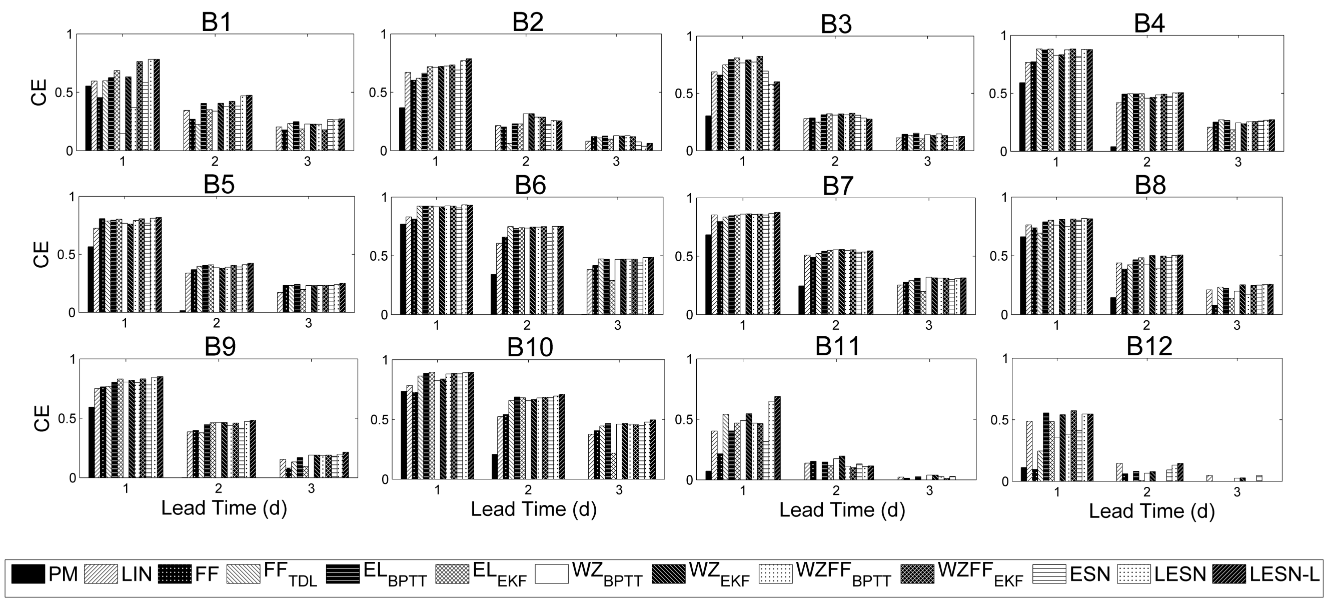

Fig. 8. Forecasting performance for lead times of 1 to 3 days. Presented are the median values of the CE over 20 model runs, cut off at minimum of 0 for clarity.

The inclusion of slower dynamics in the reservoir using leaky-integrator neurons could be beneficial to a RC R–R model, considering the broad range of time scales on which hydrological processes take place. However, the results of the so-called LESN-L model (see Figs. 4 to 7), show that the leaky-integrator neurons are not successful in significantly increasing the information content of the reservoir. Instead, the model merely ends up with a different trade-off between CE, $\mathrm{CE}_{\log}$ and MSDE, compared to the LESN model.

Figure 8 shows CE performance of all model structures for forecast lead times of 1 to 3 days. Because of the time scale of the time series (daily) in combination with the residence time of the catchments (often in the order of 12 to 24 h), the differences between models are small compared to the overall deterioration of model results. Nevertheless, the best models are again $\mathrm{EL}_{\mathrm{EKF}}$, $\mathrm{WZFF}_{\mathrm{EKF}}$ and LESN, with the latter slightly outperforming the former two. The slower dynamics inside the LESN-L model seems to give it a slightly increased capacity for forecasting over larger lead times.

5 Conclusions

Recurrent ANNs can theoretically represent river basins in a more efficient and realistic way than feedforward networks because of their intrinsic similarity (i.e. both are dynamical systems). However, such theoretical benefits do not always manifest themselves due to shortcomings of the training procedure, as exemplified by the underperformance of the popular BPTT algorithm in this work. The state-of-the-art EKF training approach, on the other hand, produced good results that prove the value of recurrent ANNs over their feedforward counterparts.

The recently introduced, conceptually simple ESN models that were the main focus of this study are found to be valid alternatives to feedforward and traditional recurrent ANNs. They show good accuracy and reliability compared to even the best traditional recurrent methods tested, often even outperforming them by a small margin. Moreover, they exhibit faster training times since this procedure is simplified to a multiple linear regression problem. By separating model dynamics and the training procedure, more insightful models are accomplished, which might benefit future research on ANNs in hydrological modelling since the lack of transparency is generally considered their biggest drawback.

However, the effectiveness of ESN models is strongly dependent on an internal network state that is both sufficiently rich and relevant to the problem at hand. For example, the standard ESN model suffered from poor performance here due to its limited instantaneous non-linear capacity. A variation with a layered reservoir that allows for a both instantaneous non-linear and dynamical mapping proved far more successful. A second, somewhat less successful reservoir modification was the use of a range of leaky-integrator neurons that enables simultaneous representation of different hydrological time scales within a reservoir. Future research that focuses on a wider range of time scales might shed more light on potential benefits of such an approach.

In conclusion, the results of this work suggest that the case for recurrency in ANN R–R models should be reconsidered. The specific architecture of ESN models overcomes several important drawbacks to traditional recurrent methods. This approach can lead to more accurate, reliable, and realistic models. Their concept of separating model dynamics simulation and network training may offer a peek into the black box of ANN R–R models, and present opportunities in hydrological modelling (e.g. as members of a model committee or in hybrid conceptual/empirical operational models).

Research on RC methods such as the ESN is still in its infancy, and more research clearly is needed on their use as R–R models, in order to further validate the technique's usefulness. Applications on different river basins, scales of space and time, and forecast horizons would increase insights into its effectiveness and reliability. Additionally, there is still a clear need for comprehensive investigations on how to maximise information content in a reservoir, and on which readouts can effectively and efficiently extract such information.

Acknowledgements. The MOPEX data set was downloaded from NOAA's National Weather Service website (http://www.nws.noaa.gov/oh/mopex/mo_datasets.htm). The ESN model results were obtained using the Reservoir Computing Toolbox for MATLAB, which is offered online by the Reservoir Lab of Ghent University (http://reslab.elis.ugent.be/). I thank the two anonymous reviewers for their constructive comments.

Edited by: H. Madsen

References

Abrahart, R. J. and See, L. M.: Comparing neural network and autoregressive moving average techniques for the provision of continuous river flow forecasts in two contrasting catchments, Hydrol. Process., 14, 2157–2172, 2000.

Abrahart, R. J., Heppenstall, A. J. and See, L. M.: Timing error correction procedure applied to neural network rainfall-runoff modelling, Hydrol. Sci. J., 52, 414–431, doi:10.1623/hysj.52.3.414, 2007.

Abrahart, R. J., Anctil, F., Coulibaly, P., Dawson, C. W., Mount, N. J., See, L. M., Shamseldin, A. Y., Solomatine, D. P., Toth, E. and Wilby, R. L.: Two decades of anarchy? Emerging themes and outstanding challenges for neural network river forecasting, Prog. Phys. Geog., 36, 480–513, doi:10.1177/0309133312444943, 2012a.

Abrahart, R. J., Mount, N. J. and Shamseldin, A. Y.: Discussion of "Reservoir Computing approach to Great Lakes water level forecasting" by P. Coulibaly [J. Hydrol. 381 (2010) 76–88]. J. Hydrol., 422–423, 76–80, doi:10.1016/j.jhydrol.2011.10.006, 2012b.

Anctil, F., Michel, C., Perrin, C., and Andréassian, V.: A soil moisture index as an auxiliary ANN input for stream flow forecasting, J. Hydrol., 286, 155–167, 2004.

Atiya, A. F. and Parlos, A. G.: New results on recurrent network training: unifying the algorithms and accelerating convergence, IEEE T. Neural Netw., 11, 697–709, 2000.

Barreto, G. de A. and Araújo, A. F. R.: A self-organizing NARX network and its application to prediction of chaotic time series, in: Proceedings of the IEEE Intl. Joint Conference on Neural Networks, vol. 3, Washington D.C., USA, 2144–2149, 2001.

Bengio, Y., Simard, P., and Frasconi, P.: Learning long-term dependencies with gradient-descent is difficult, IEEE T. Neural Netw., 5, 157–166, 1994.

Beven, K. J., Lamb, R., Quinn, P. F., Romanowicz, R., and Freer, J.: TOPMODEL, in: Computer Models of Watershed Hydrology, edited by: Singh, V. P., Water Resources Publications, Colorado, 627–668, 1995.

Brazil, L. E.: Multilevel calibration strategy for complex hydrologic simulation models, Ph.D. dissertation, Colo. State Univ., Fort Collins, 1988.

Buehner, M. and Young, P.: A tighter bound for the echo state property, Neural Netw., 17, 820–824, 2006.

Burnash, R. J. C.: The NWS river forecast system – catchment modeling, in: Computer Models of Watershed Hydrology, edited by: Singh, V. P., Water Resources Publications, Colorado, 311–366, 1995.

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

Chang, F. J., Chiang, L. C., and Huang, H. L.: Real-time recurrent learning network for stream-flow forecasting, Hydrol. Process., 16, 2577–2588, 2002.

Chiang, Y. M., Chiang, L. C., and Chang, F. J.: Comparison of static-feedforward and dynamic-feedback neural networks for rainfall-runoff modeling, J. Hydrol., 290, 297–311, 2004.

Coulibaly, P.: Reservoir computing approach to Great Lakes water level forecasting, J. Hydrol., 381, 76–88, 2010.

Coulibaly, P., Anctil, F., Rasmussen, P., and Bobée, B.: A recurrent neural networks approach using indices of low-frequency climatic variability to forecast regional annual runoff, Hydrol. Process., 14, 2755–2777, 2000.

Cunge, J. A.: Of data and models, J. Hydroinform., 5, 75–98, 2003.

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.

de Vos, N. J. and Rientjes, T. H. M.: Correction of timing errors of artificial neural network rainfall-runoff models, in: Practical Hydroinformatics, edited by: Abrahart, R. J., See, L. M., and Solomatine, D. P., Water Science and Technology Library, Springer, 2008a.

de Vos, N. J. and Rientjes, T. H. M.: Multi-objective training of artificial neural networks for rainfall-runoff modeling, Water Resour. Res., 44, W08434, doi:10.1029/2007WR006734, 2008b.

Doya, K.: Bifurcations in the learning of recurrent neural networks, in: Proc. IEEE Int. Symposium on Circuits and Systems, vol. 6, San Diego, CA, USA, 2777–2780, 1992.

Duan, Q., Schaake, J., Andréassian, V., Franks, S., Goteti, G., Gupta, H. V., Gusev, Y. M., Habets, F., Hall, A., Hay, L., Hogue, T., Huang, M., Leavesley, G., Liang, X., Nasonova, O. N., Noilhan, J., Oudin, L., Sorooshian, S., Wagener, T., and Wood, E. F.: Model Parameter Estimation Experiment (MOPEX): an overview of science strategy and major results from the second and third workshops, J. Hydrol., 320, 3–17, 2006.

Elman, J. L.: Finding structure in time, Cognitive Sci., 14, 179–211, 1990.

Hagan, M. T. and Menhaj, M. B.: Training feedforward networks with the Marquardt algorithm, IEEE T. Neural Netw., 5, 989–993, 1994.

Hammer, B., Schrauwen, B., and Steil, J. J.: Recent advances in efficient learning of recurrent networks, in: European Symposium on Artificial Neural Networks, Bruges, Belgium, 213–226, 2009.

Haykin, S.: Neural Networks, a Comprehensive Foundation, Prentice Hall, Upper Saddle River, 1999.

Haykin, S.: Kalman Filtering and Neural Networks, Wiley, New York, 2001.

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

Hsu, K.-L., Gupta, H. V., and Sorooshian, S.: Application of a recurrent neural network to rainfall-runoff modeling, in: 24th Annual Water Resources Planning and Management Conference, Houston, TX, USA, 68–73, 1997.

Jaeger, H.: The echo state approach to analysing and training recurrent neural networks, Tech. Report GMD Report 148, German National Research Center for Information Technology, St. Augustin, Germany, 2001.

Jaeger, H.: A tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the “echo state network” approach, Tech. Report GMD Report 159, German National Research Center for Information Technology, St. Augustin, Germany,, 2002.

Jaeger, H. and Haas, H.: Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication, Science, 304, 78–80, 2004.

Jain, A. and Srinivasulu, S.: Development of effective and efficient rainfall-runoff models using integration of deterministic, real-coded genetic algorithms and artificial neural network techniques, Water Resour. Res., 40, W04302, doi:10.1029/2003WR002355, 2004.

Jang, J.-S., R.: ANFIS: Adaptive-network-based fuzzy inference systems, IEEE T. Syst. Man. Cyb., 23, 665–685, 1993.

Lindström, G., Johansson, B., Persson, M., Gardelin, M., and Bergström, S.: Development and test of the distributed HBV-96 hydrological model, J. Hydrol., 201, 272–288, 1997.

Lukoševičius, M.: Echo state networks with trained feedbacks, Tech. Report No. 4, Jacobs University Bremen, 2007.

Lukoševičius, M. and Jaeger, H.: Reservoir computing approaches to recurrent neural network training, Comput. Sci. Rev., 3, 127–149, 2009.

Maass, W., Natschläger, T., and Markram, H.: Real-time computing without stable states: a new framework for neural computation based on perturbations, Neural Comput., 14, 2531–2560, 2002.

Møller, M. F.: A scaled conjugate gradient algorithm for fast supervised learning, Neural Netw., 6, 525–533, 1993.

Puskorius, G. V. and Feldkamp, L. A.: Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks, IEEE T. Neural Netw., 5, 279–297, 1994.

Rumelhart, D. E. and McLelland, J. L.: Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1, MIT Press, Cambridge, 1986.

Saad, E. W., Prokhorov, D. V., and Wunsch, I.: Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks, IEEE T. Neural Netw., 9, 1456–1470, 1998.

Sacchi, R., Carneiro, A. A. F. M., and Araújo, A. F. R.: A RBF network trained by the SONARX model and applied to obtain the operation policies of the hydropower systems, in: Brazilian Symposium on Neural Networks – SBRN, Brazil, 2004.

Sacchi, R., Ozturk, M. C., Principe, J. C., Carneiro, A. A. F., and da Silva, I. N.: Water inflow forecasting using the Echo State Network: a Brazilian case study, in: Proceedings of the IEEE Intl. Joint Conference on Neural Networks, Orlando, FL, USA, 2403–2408, 2007.

Shamseldin, A. Y.: Application of a neural network technique to rainfall-runoff modelling, J. Hydrol., 199, 272–294, 1997.

Steil, J. J.: Backpropagation-decorrelation: recurrent learning with O(N) complexity, in: Proceedings of the IEEE Intl. Joint Conference on Neural Networks, vol. 2, Budapest, Hungary, 843–848, 2004.

Sum, J., Chan, L., Leung, C., and Young, G.: Extended Kalman filter-based pruning method for recurrent neural networks, Neural Comput., 10, 1481–1506, 1998.

Verstraeten, D., Schrauwen, B., D'Haene, M., and Stroobandt, D.: An experimental unification of reservoir computing methods, Neural Netw., 20, 391–403, 2007.

Werbos, P. J.: Backpropagation through time: what it does and how to do it, Proc. IEEE, 78, 1550–1560, 1990.

Williams, R. J. and Zipser, D.: A learning algorithm for continually running fully recurrent neural networks, Neural Comput., 1, 270–280, 1989.

Wyffels, F., Schrauwen, B., and Stroobandt, D.: Stable output feedback in reservoir computing using ridge regression, in: Proc. 18th Int. Conference on Artificial Neural Networks, vol. 5163, Prague, Czech Republic, 808–817, 2008.

Yonaba, H., Anctil, F., and Fortin, V.: Comparing sigmoid transfer functions for neural network multistep ahead streamflow forecasting, J. Hydrol. Eng., 15, 275–283, 2010.