PUBLISHED VERSION

Bowden, Gavin James; Maier, Holger R.; Dandy, Graeme Clyde
Real-time deployment of artificial neural network forecasting models: Understanding the range of applicability, *Water Resources Research*, 2012; 48(10):W10549.

Copyright 2012. American Geophysical Union. All Rights Reserved.

PERMISSIONS

http://publications.agu.org/author-resource-center/usage-permissions/

Permission to Deposit an Article in an Institutional Repository

Adopted by Council 13 December 2009

AGU allows authors to deposit their journal articles if the version is the final published citable version of record, the AGU copyright statement is clearly visible on the posting, and the posting is made 6 months after official publication by the AGU.

8th October 2013

http://hdl.handle.net/2440/75444

Real-time deployment of artificial neural network forecasting models: Understanding the range of applicability

Gavin J. Bowden,[1] Holger R. Maier,[1] and Graeme C. Dandy[1]

Received 12 February 2012; revised 31 August 2012; accepted 25 September 2012; published 31 October 2012.

[1] When an operational artificial neural network (ANN) model is deployed, new input patterns are collected in order to make real-time forecasts. However, ANNs (like other empirical and statistical methods) are unable to reliably extrapolate beyond the calibration range. Consequently, when deployed in real-time operation there is a need to determine if new input patterns are representative of the data used in calibrating the model. To address this problem, a novel detection system for identifying uncharacteristic data patterns is presented. This approach combines a self-organizing map (SOM), to partition the data set, with nonparametric kernel density estimators to calculate local density estimates (LDE). The SOM-LDE method determines the degree to which a new input pattern can be considered to be contained within the domain of the calibration set. If a new pattern is found to be uncharacteristic, a warning can be issued with the forecast, and the ANN model retrained to include the new pattern. This approach of selectively retraining the model is compared to no retraining and the more computationally onerous case of retraining the model after each new sample. These three approaches are applied to forecast flow in the Kentucky River, USA, using multilayer perceptron (MLP) models. The results demonstrate that there is a significant advantage in retraining an ANN that has been deployed as a real-time, operational model, and that the SOM-LDE classifier is an effective approach for identifying the model's range of applicability and assessing the usefulness of the forecast.

Citation: Bowden, G. J., H. R. Maier, and G. C. Dandy (2012), Real-time deployment of artificial neural network forecasting models: Understanding the range of applicability, *Water Resour. Res.*, *48*, W10549, doi:10.1029/2012WR011984.

1. Introduction

[2] Artificial neural networks (ANNs) have risen to prominence as a viable alternative to many traditional water resources models, particularly in the field of forecasting hydrologic variables. Some of the important features that have contributed to their popularity include their ease of implementation, their ability to learn from examples without explicit knowledge of the underlying physics, and their powerful generalization abilities. ANNs have been applied to a wide variety of water resources problems including rainfall-runoff modeling, precipitation forecasting, streamflow forecasting, and groundwater and water quality modeling [*ASCE Task Committee on Application of Artificial Neural Networks in Hydrology*, 2000; *Maier and Dandy*, 2000; *Maier et al.*, 2010; *Abrahart et al.*, 2012]. However, one limitation of ANNs is that, like other empirical methods, they are unable to reliably extrapolate beyond the range of the data used for calibration [*Flood and Kartam*, 1994; *Minns and Hall*, 1996; *Tokar and Johnson*, 1999]. This well-known limitation of data-driven models is primarily because they are not based on the underlying physics. Physically based models tend to perform better at model extrapolation for inputs that are outside of the range of those used in the calibration data as the mass and energy constraints they comply with may still result in an appropriate response. Accordingly, it can be very difficult to determine when data-driven models, such as ANNs, will fail to generalize and to understand the range of applicability of the model.

[3] It has been acknowledged in the past that an ANN is susceptible to becoming "...a prisoner of its training data" [*Minns and Hall*, 1996]. During prediction, the model is likely to perform poorly if faced with a set of inputs that is far removed from the examples that it saw during training. By using the widest limits of examples during training provides the best chance of avoiding the need to extrapolate with an ANN [*ASCE Task Committee on Application of Artificial Neural Networks in Hydrology*, 2000]. Recent research into data splitting has endeavored to determine a systematic methodology for creating representative calibration subsets that reduce uncertainty due to sampling variance [*May et al.*, 2010]. However, once a trained ANN model has been deployed in an operational setting, it is still likely to perform poorly if faced with new input patterns that are far removed from the examples that it was presented with during calibration. This problem led the *ASCE Task Committee on Application of Artificial Neural Networks in Hydrology* [2000] to pose the following question: "Very often we may have no alternative but to proceed

[1]School of Civil, Environmental and Mining Engineering, University of Adelaide, Adelaide, South Australia, Australia.

Corresponding author: H. R. Maier, School of Civil, Environmental and Mining Engineering, University of Adelaide, Adelaide, SA 5005, Australia. (holger.maier@adelaide.edu.au)

Published in 2012 by the American Geophysical Union.

with limited data. Under these circumstances can we say when generalization will fail so that we understand the range of applicability of the ANN?" Once a model has been calibrated and deployed, this equates to knowing when the model is likely to fail and when the model needs to be recalibrated to incorporate new, uncharacteristic patterns that have not been included in the data set that was used to calibrate the model in the first place.

[4] Representative data can be viewed in a number of ways, including, a model's ability to generate state and output variables. A data pattern may not be contained within the calibration data but may still be capable of generating a suitable output and hence, could be considered to be representative. However, in real-time forecasting applications it is difficult to know whether inputs are generating suitable outputs or not. In such cases, we are restricted to comparing our current inputs with the inputs used during calibration. In this context, representative data can be considered to be patterns that are contained within the domain of the calibration set. Similar definitions have been provided in the studies conducted by *Bowden et al.* [2002] and *May et al.* [2010], who investigated methods of improving ANN performance through optimal allocation of data to the training, testing, and validation sets. Both studies have shown that model performance is degraded when unrepresentative data are encountered in this context.

[5] In general, there are two main approaches that have been employed in previous studies to improve the generalization ability of ANNs in the prediction of water resources variables. The first approach seeks to improve the extrapolation ability of ANNs through improved calibration methods and the second involves updating ANN model parameters in real-time during deployment. A number of authors have investigated the extrapolation ability of ANNs and considered various approaches to help improve generalization, with varying degrees of success. Examples of such studies include *Cigizoglu* [2003], *Coulibaly et al.* [2000, 2001], *Giustolisi and Laucelli* [2005], *Hu et al.* [2005], and *Imrie et al.* [2000]. There have also been many publications that have explored the use of ANNs in real-time forecasting, including *Aquil et al.* [2007], *Brath et al.* [2002], *Bruen and Yang* [2005], *Chang et al.* [2002], *Chen and Yu* [2007], *Goswami et al.* [2005], *Goswami and O'Connor* [2007], *Napolitano et al.* [2010], and *Xiong and O'Connor* [2002]. However, to the authors' knowledge, previous studies in water resources applications have not addressed the issue of how to systematically detect when ANN generalization is likely to fail and, consequently, when the deployed ANN model needs to be recalibrated.

[6] When an ANN model is deployed in a real-time, operational setting, there are three options available to the practitioner: (1) no recalibration, i.e., keeping the model's parameters fixed, (2) recalibration of the model at some arbitrary time interval, and (3) recalibration of the model given some knowledge of when a pattern is encountered that is not sufficiently represented in the calibration domain. The first two options are straightforward, however, the last option requires a method for detecting data that are not representative of the calibration data set, and as such, is a closely related problem to that of detecting outliers in multivariate space.

[7] *Hawkins* [1980] provides the following generally accepted definition of an outlier: "An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism." Accordingly, in outlier detection, the task is to identify patterns far removed from the majority of observations. In this investigation, the task is to identify uncharacteristic data patterns as they arise in a real-time setting. These are defined as patterns that are not sufficiently represented in the calibration data. Therefore, the aim of this work is to develop a suitable technique that can be used in real-time to identify new patterns that are outside the domain of the calibration data and to then recalibrate the model once these patterns have been identified. These data patterns may or may not be outliers in the conventional sense but are referred to as uncharacteristic data in this study since they are not sufficiently represented by the patterns contained in the original calibration set.

[8] *Bowden et al.* [2002] proposed a clustering method for diagnosing uncharacteristic data patterns in a multivariate space using a self-organizing map (SOM). It was found that by combining new, unseen data with the calibration data and clustering these data using a SOM, regions of poor performance could be identified by examining the resulting clusters. If the new data formed a cluster that did not contain any training data, then these data were diagnosed as uncharacteristic. It was found that the ANN model performed poorly on these uncharacteristic data since it had not considered these events during training. To determine when the ANN is extrapolating rather than interpolating, it is necessary to know the distribution of the calibration data, however, this can be rather difficult to determine when the ANN model has a large number of inputs. One way to address this problem is by plotting histograms of the inputs in the training set, to see which values are most common and which values are rare or absent from the training set. But this is somewhat subjective and becomes increasingly difficult as the number of inputs increases.

[9] In this paper the method proposed by *Bowden et al.* [2002] is extended to create a robust and systematic detection system that can be used to identify data patterns that are not sufficiently represented by the calibration data. A classifier consisting of two important components is developed to detect uncharacteristic data. The first component is a SOM that provides a two-dimensional topological mapping of the multivariate calibration data and partitions these data into a number of regions. It is then possible to use the SOM to define the neighborhood of calibration data most closely matching any new input patterns encountered during real-time operation. The second component of the classifier is a kernel density estimator, which provides local density estimates (LDEs). To identify uncharacteristic data patterns, the local density of each new pattern is compared to the local density of the neighboring calibration data identified by the SOM and a local uncharacteristic factor (LUF) is then calculated. When a new input pattern is found to be uncharacteristic, there is likely to be a significant degree of uncertainty associated with the corresponding forecast and consequently, a warning is issued. This pattern is then added to the existing calibration data. Once the corresponding output has been collected, which might take some time, depending on the modeling time step and forecasting period adopted, the ANN is recalibrated with this pattern included. In this way, the ANN model is able to adapt to

new information as it is encountered and becomes increasingly robust with time. This method also has the advantage that if an uncharacteristic pattern is detected, a warning can be issued in real-time with the corresponding forecast.

[10] To assess the efficacy of the proposed approach, it is applied to a real-world case study. This involves forecasting flow based on rainfall and previous streamflow using data from the Kentucky River basin. The dynamic and nonlinear rainfall-runoff process provides a good test of these techniques and a 13 year real-time simulation is performed using calibration sets of various lengths to assess the utility of actively retraining the model. The data set for this case study was compiled by *Jain and Srinivasulu* [2006] to build ANN models and this data set was provided for the investigation performed in this paper. Three recalibration scenarios are investigated for this case study, namely: (1) no recalibration, (2) recalibration after each new data sample is collected, and (3) recalibration when an uncharacteristic pattern is detected using the proposed SOM-LDE classifier.

2. Proposed Recalibration Methodology

2.1. Conceptual Approach

[11] During ANN model development, the available data are generally divided into training, testing, and validation subsets. The training set is used to determine the ANN's connection weights, the testing set is used to decide when to stop training in order to avoid overfitting (holdout method) and/or which network architecture is most suitable, and the validation set is used to assess the trained ANN's generalization ability. Since ANNs are unreliable at extrapolation, all of the patterns present in the available data need to be represented in the training set. Likewise, since the testing set is used to decide when to stop training and which network structure is optimal, it also must be representative of the training set and should also contain all of the patterns present in the available data. If all available patterns are used to calibrate the model, then the most rigorous test of the model's generalization ability would be if all of the types of patterns are also represented in the validation set. However, in many applications the data available to the practitioner at the time of model development are not likely to contain all possible patterns, especially when the available data span a relatively short record. Hydrologic systems are also dynamic and can evolve over time, exhibiting a high degree of nonstationarity. Consequently, future events that are caused by different drivers and processes are unlikely to be adequately represented by the calibration data. An example of such a situation may occur when considering the changes to a hydrologic system caused by climate change impacts. In such cases, once an ANN model has been deployed in a real-time setting, it is necessary to ensure that the performance of the model will not deteriorate significantly when exposed to future input/output data patterns that were not included in the original model development process.

[12] The proposed conceptual approach for achieving this level of adaptation is based on recalibration, as this ensures that all new data patterns are incorporated into the model, thereby extending the model's range of applicability over time. However, rather than recalibrating the model every time a new input set is presented to the model, which could require significant computing time, it is proposed to conduct a check to see if the data that have been used for model calibration thus far are representative of the input data used during real-time deployment. If this is the case, there is no need to recalibrate the model. However, if the new input vector is significantly different from the patterns contained in the calibration data, then it is labeled as uncharacteristic and should be added to the calibration set, and the model recalibrated. The advantages of this approach are that (1) it is possible to detect and warn when the forecast may contain a large degree of uncertainty and (2) the range of applicability of the ANN model is defined. Recalibrating only upon detecting an uncharacteristic data pattern also has the advantage of reducing the computational burden, which is especially beneficial in high frequency forecasting applications. However, it should be noted that there is some computational effort involved in the check to determine whether the original calibration data are representative of the current input pattern. Also, recalibration would only be possible in a real-time application where there was sufficient time between forecasts in order to perform the recalibration. For very high frequency applications it may only be possible to do an online weight update rather than a full recalibration of the model.

[13] The detailed conceptual approach to model recalibration is shown in Figure 1. The process starts after model development has been completed ($t = 0$). Once the model has been deployed in an operational setting, the time counter t is updated by one increment once a new input vector becomes available, and the model is used to obtain a prediction/forecast. Next, the new input vector (input vector t) is added to the input data currently contained in the calibration set (calibration data $t - 1$). The multidimensional data space consisting of the combined input data is then partitioned into representative regions. The regions are used to define a neighborhood surrounding the new input vector (input vector t) and a local probability density based on the data points in the neighborhood is calculated. A statistical significance test is then implemented to compare the local density of each new input vector with the local density of its calibration data neighbors and thereby define a local uncharacteristic factor (LUF). If the new input vector is representative of the calibration data, the predictive performance of the model is likely to be adequate, as the model output is obtained by interpolation. Consequently, the current model can be used for predictive purposes and there is no need to add the new input vector to the calibration set. However, if the new input vector is located in a region of the input space that is different from those occupied by the input vectors in the original calibration set, the latter are not representative of the former and the predictive performance of the model is likely to be compromised. While the model can still be used to obtain the desired prediction/forecast, a warning should be issued that the model outputs should be treated with caution. When using a Bayesian ANN model, this increased uncertainty will be reflected in the spread of the prediction limits [see *Kingston et al.*, 2005].

[14] Input vectors that represent a different pattern from those contained in the training set should be added to the calibration data and the model recalibrated to broaden its range of applicability to include this new pattern. However,

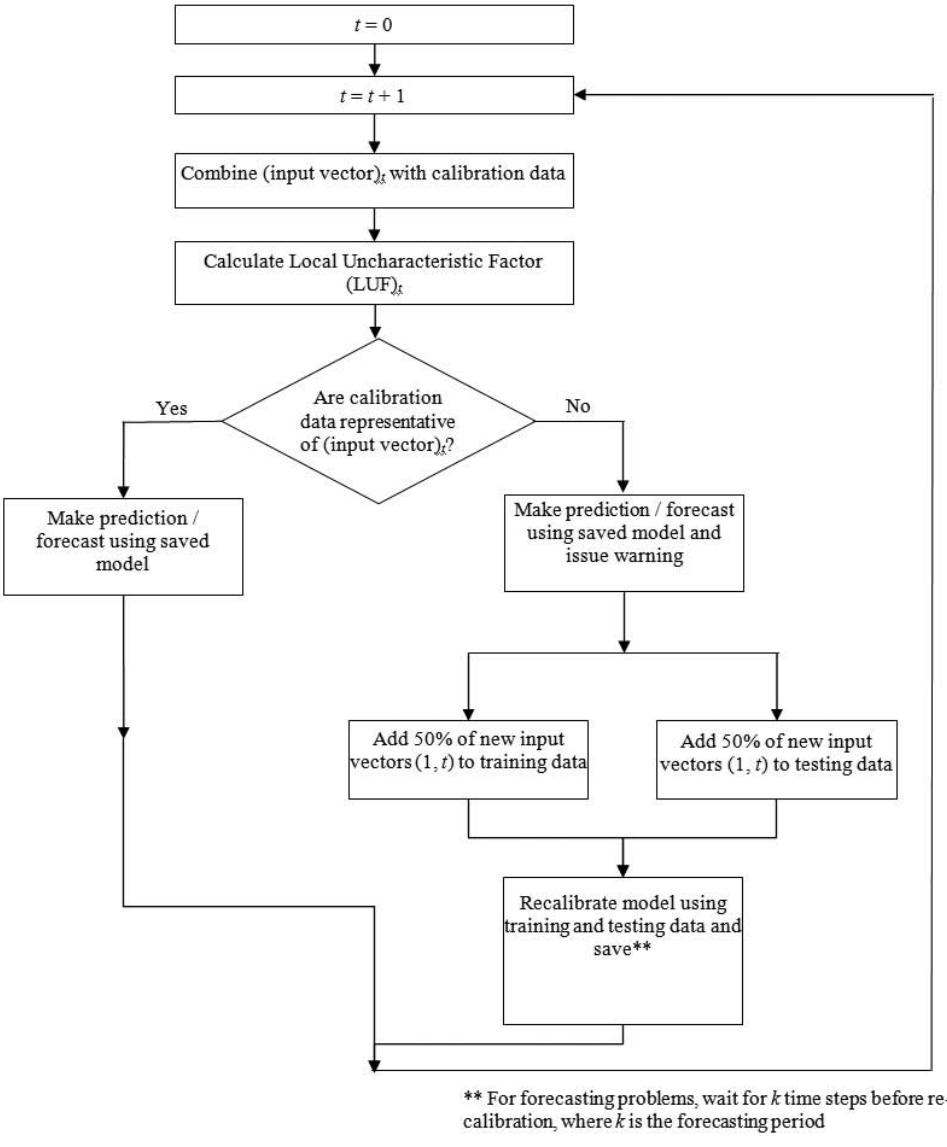

Figure 1. Conceptual approach to model recalibration.

before the model can be recalibrated, the measured output data corresponding to the new input vector have to be obtained. This may cause a delay in the recalibration process, particularly if the model is used to obtain multistep ahead forecasts, as the output data will only become available after the forecasting period has elapsed (e.g., if the purpose of the model is to produce a seven day forecast, the measured output will only become available in seven days). In addition, in cases where a test set is being used, care needs to be taken to ensure that the new input patterns are distributed between the calibration and test sets. Care also needs to be taken to ensure the model does not become overrepresented by a particular type of model pattern. As shown in Figure 1, for the sake of simplicity of implementation, it is proposed that when an unrepresentative input vector is detected, 50% of all new input patterns encountered since the last model calibration should be added to the training set, and the remaining 50% allocated to the test set.

2.2. Detecting Uncharacteristic Data

[15] The problem of identifying uncharacteristic data patterns is closely related to outlier detection. Outlier detection is one of the most important tasks in data analysis and there are many methods that have been proposed for achieving this. There are four general approaches to the problem, although in practice there is usually considerable overlap between them. The approaches can be categorized as: (1) statistical, (2) distance-based, (3) clustering, and (4) model-based.

[16] Statistical approaches for outlier detection generally make use of a probabilistic data model and data points are determined to be outliers if they do not fit the distributional model [*Petrovskiy*, 2003]. In most cases, estimates of location and shape are required, but these are susceptible to the problem of outliers masking each other, and this is further exacerbated for multidimensional data vectors [*Nag et al.*, 2005]. In distance-based methods, metrics such as the

Euclidean [*Knorr and Ng*, 1998] or Mahalanobis distance [*Atkinson*, 1994] are commonly used to calculate distances between points. These can be further divided into methods that compute (1) the full dimensional distance, either using all available features, or only feature projections [e.g., *Knorr and Ng*, 1998], and (2) the densities of local neighborhoods [e.g., *Breunig*, 2000]. Since a probabilistic interpretation can be placed on many distance-based methods, there is overlap between this approach and statistical approaches. Clustering methods [e.g., *Nag et al.*, 2005] have also been widely used to detect outliers, either as points that do not belong to clusters, i.e., that form singleton clusters, or as clusters that are significantly smaller than others [*Latecki et al.*, 2007]. Finally, model-based methods make use of a predictive model to describe typical behavior and outliers are then detected as deviations from the learned model [e.g., *Hawkins et al.*, 2002; *Tax and Duin*, 1998].

[17] The approach proposed in this study uses a combination of clustering and distance-based methods. It is not possible to use density estimates to detect outliers for multimodal distributions, which are often encountered in real-world, multidimensional data sets. This is because data points that belong to different modal components may have different densities without necessarily being outliers [*Latecki et al.*, 2007]. To overcome this, a self-organizing map (SOM) has been used in this study to partition the data set and define the local neighborhood surrounding the data point to be queried. In order to avoid making any assumptions about the underlying data, nonparametric kernel density estimation is used to estimate the local densities.

[18] The ability to partition the calibration data into clusters and then consider the local distribution has been shown to be advantageous by *May et al.* [2010], who provided evidence that the overall ANN model error was dependent on the accuracy of the predictions within local regions. Accordingly, the data splitting methods that were able to consider the local distribution of the data were more effective in obtaining representative training, testing, and validation data sets. By extension, this is also important in the case of model deployment since it is the local distribution of training data surrounding the new input pattern that governs how well the prediction model is likely to perform for that pattern.

2.2.1. Self-Organizing Map (SOM) Clustering

[19] The SOM was developed by *Kohonen* [1982] and arose from attempts to develop topographically organized maps of multidimensional data. The SOM is able to partition a data set by using an n-dimensional array of processing elements that learn the optimal distribution of the weight vectors. In this way, input data, which may have many dimensions, can come to be represented by a one- or two-dimensional vector which preserves the order of the higher dimensional data and provides a nonparametric estimation of the underlying distribution.

[20] The SOM has been widely employed in water resources applications. A recent study by *Adeloye et al.* [2011] used a SOM to predict the reference crop evapotranspiration based on daily weather data at two diverse basins. It was found that the SOM-based approach provided estimates that were in good agreement with conventional methods, despite requiring fewer input variables. *Pearce et al.* [2011] used a modified SOM as part of a decision making methodology to characterize a water quality gradient in leachate—contaminated groundwater using only microbiological data for input. The SOM was modified by weighting the input variables by their relative importance and provided statistical guidance for classifying sample similarities. *Fassnacht and Derry* [2010] used a SOM to define regions of homogeneity in the Colorado River Basin using snow telemetry (SNOTEL) snow water equivalent data. *Sahoo and Ray* [2008] presented an approach that utilized a SOM to identify which samples, and how many, to include in training, testing and validation sets for optimum ANN prediction efficiency. A genetic algorithm (GA) was used to optimize the model structure and parameters and it was found that the GA-ANN model using the SOM technique for data division was able to outperform the GA-ANN using arbitrary division. *Parasuraman et al.* [2006] used a SOM as the spiking layer to a modular neural network (SMNN) and applied the model to two case studies, including streamflow forecasting and the modeling of eddy covariance-measured evapotranspiration. It was found that the SMNN models outperformed conventional feedforward ANNs for both case studies as they were very effective in discretizing the complex mapping space into simpler domains that could be learned with relative ease.

[21] Details of the SOM algorithm used in the present research are provided by *Bowden et al.* [2005b]. In most applications, the size of the SOM grid is chosen by a trial-and-error process. However, for ease of implementation, the approach adopted in this study is to use the heuristic rule provided by *Vesanto* [1999] for determining the size of the SOM grid based on the number of samples to be clustered. The number of grid cells is given by

$$m = \beta n^{0.54}, \tag{1}$$

where values of 0.2, 1, and 5 are used for the constant β for small, normal and large SOMs, respectively, and n is the number of data patterns [*Vesanto*, 1999].

[22] The dimensions of the grid are also important for the quality of the mapping as it has been observed that an $r \times c$ SOM with one side of greater length than the other (i.e., rectangular) is superior to a square $r \times r$ SOM since the former is more easily able to provide a mapping if the data are distributed along a dominant axis [*May et al.*, 2010]. Therefore, given a ratio of the SOM dimensions γ it is possible to formulate the dimensions of the SOM in terms of the number of rows, by considering $r = \gamma c$. Since $m = rc$, the number of grid cells can then be written, in terms of r, as

$$m = \frac{r^2}{\gamma} \tag{2}$$

which can be substituted into (1) to give the number of SOM rows as

$$r = \sqrt{\gamma \beta n^{0.54}}. \tag{3}$$

[23] The rule presented in (3) was investigated by *May et al.* [2010] and was found to provide adequate clustering for the data used in their case studies, without requiring the trial-and-error evaluation of clustering with a potentially

large number of SOMs with different dimensions. Consequently, the heuristic rule provided in equation (3) was adopted in this study and values of $\beta = 1$ and $\gamma = 1.6$ were used in accordance with the findings of *May et al.* [2010]. Initial trials showed that the results obtained in this study were not sensitive to these SOM sizing parameters.

2.2.2. Kernel Density Estimation

[24] One of the most common forms of nonparametric density estimation is the kernel density estimator, which for n data samples of dimensionality d is able to provide an estimate of the distribution density (4). Kernel density estimation techniques are widely used due to their stability, efficiency, and robustness.

$$\hat{f}_X(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\lambda^d}K\left(\frac{x - x_i}{\lambda}\right), \tag{4}$$

where $K(\cdot)$ is the kernel function (satisfying non-negativity and normalization conditions) and $\lambda > 0$ is the smoothing parameter, called the bandwidth. In most applications, $K(\cdot)$ is a Gaussian kernel function of dimensionality d with zero mean and unit standard deviation.

[25] In order to produce an accurate density estimate, the bandwidth λ must be chosen very carefully. Small values of λ tend to lead to density estimates that give too much emphasis to individual points. In this case, spurious fine structure may become present with bumps in the tails of the probability density. Larger values of λ tend to oversmooth the probability density with all detail, spurious or otherwise, becoming obscured. Only for intermediate values of λ will a good density estimate be obtained. Various rules-of-thumb are available in the literature to help choose an optimal value of λ. *Sharma* [2000] used the Gaussian reference bandwidth [*Scott*, 1992], as it is relatively simple and computationally efficient. *Bowden et al.* [2005a] found that kernel density estimators using the city-block distance were more computationally efficient, yet still able to provide equivalent density estimations when compared to estimators using the Gaussian kernel. Consequently, the city block distance kernel function has been used in this research. Since the objective of this paper is to detect uncharacteristic data patterns based on comparing them with samples in their local neighborhood, the summation only needs to be taken over the neighborhood defined by the SOM. If $kN_c(x)$ denotes the k neighbors of a sample x, then by combining this with the city-block distance kernel, it is possible to obtain the following equation for estimating the local density at a query data pattern x_Q:

$$\begin{aligned}\hat{f}_X(x_Q) &= \frac{1}{k(2\lambda)^d}\sum_{x_i \in kN_c(x_Q)}\prod_{j=1}^{d} e^{-\frac{|x_j - x_{ij}|}{\lambda}} \\ &= \frac{1}{k(2\lambda)^d}\sum_{x_i \in kN_c(x_Q)}\exp\left(-\frac{1}{\lambda}\sum_{j=1}^{d}|x_j - x_{ij}|\right),\end{aligned} \tag{5}$$

where $\hat{f}_X(x_Q)$ is the multivariate kernel density estimate of the d-dimensional variable set X at coordinate location x_Q, x_i is the ith multivariate data point, for a sample of size n points within the query patterns' neighborhood, and λ is the bandwidth of the kernel density estimate. The univariate version of this kernel was first proposed by *Parzen* [1962]. However, theorem 4.1 in *Cacoullous* [1966] shows that Parzen's results can be extended to the multivariate case when the multivariate kernel is a product of a number of univariate kernels, as is the case in (5). In (5), j indexes the product of kernels calculated over the d-dimensional data pattern. The second line of equation (5) shows the equivalent function transformed into a summation of the city-block distance over the d-dimensional vector. This kernel estimator is particularly attractive from the point of view of computational simplicity. Via experimentation, it was determined that a neighborhood size of $k = 30$ provided the most stable results when identifying uncharacteristic data patterns. Having at least 30 neighboring samples in the reference population was considered necessary for ensuring a statistically representative population.

2.2.3. Local Uncharacteristic Factor (LUF)

[26] By comparing the density of the query pattern with the densities of the neighboring data points in the training set determined by the SOM, it is possible to define a local uncharacteristic factor (LUF). This measure provides the degree to which a new data pattern can be considered to be uncharacteristic, based on its neighborhood of calibration data. The LUF, introduced in this paper, is defined as the ratio of the average local density estimate (LDE) of the data patterns in the neighborhood surrounding the new data pattern to the new pattern's LDE:

$$LUF(x_j) = \frac{\sum_{x_i \in kN_c(x_Q)}\frac{\hat{f}_X(x_i)}{k}}{\hat{f}_X(x_Q)}, \tag{6}$$

where $\hat{f}_X(x_Q)$ is the local density estimate of new query pattern x_Q and $\hat{f}_X(x_i)$ are the density estimates of the k calibration patterns in x_Q's neighborhood N_c. The LUF of a new data pattern x_Q captures the degree to which it is deemed to be unrepresentative of the calibration data. Based on this relationship, uncharacteristic data patterns can be identified by setting a criterion of the form $LUF(x_Q) > T$, where T is a threshold for deciding if a pattern is uncharacteristic relative to its local calibration data. The selection of T is case study dependent and will vary based on the local fluctuations and clustering of the given data set. However, it is important here to err on the side of caution and set T at a relatively conservative value since the aim is to detect the patterns that are poorly represented by the calibration data. There is a trade-off between detecting too few uncharacteristic patterns (false negatives) if T is too large and detecting too many uncharacteristic patterns (false positives) if T is too small, which also has the added disadvantage of increasing computational overhead due to excessive retraining of the model. It was determined by trial-and-error that a value of $T \approx 2$ provided a suitable compromise for detecting poorly represented data patterns in the case study investigated in this paper, and therefore, this value was adopted. The trials conducted showed that the classifier had good performance across a very broad range of values for T and, hence, the method was not sensitive to this parameter. Performance only started to degrade for values of $T > 10{,}000$. A value of $T \approx 2$ provided good classification performance while still only requiring the model to be retrained a small percentage of the time.

3. Case Study

[27] As mentioned previously, the purpose of the real-world case study is to assess the performance of three approaches for deploying ANN models in a real-time, operational setting, including: (1) no retraining (i.e., the model parameters remained fixed), (2) always retraining the model every time a new input/output set becomes available, and (3) selectively retraining the model based on the SOM-LDE classifier identifying a new pattern that is not sufficiently represented by the calibration data. The real-world study presented here is a rainfall-runoff forecasting application.

3.1. Study Area and Data

[28] For this investigation, the case study and data set described by *Jain and Srinivasulu* [2006] were used. This data set was derived from the Kentucky River basin. In total, 26 years of daily data (1960–1972 and 1977–1989) were available and include the average daily streamflow (cubic feet per second) from the Kentucky River at LD10, and the average daily rainfall (mm) from five rain gauges (Manchester, Hyden, Jackson, Heidelberg, and Lexington Airport) scattered throughout the Kentucky River catchment. *Jain and Srinivasulu* [2006] transformed the precipitation data into effective rainfall using the infiltration modeling and soil moisture accounting (SMA) procedure described in *Jain et al.* [2004]. Consequently, the effective rainfall data were also used as part of this investigation.

3.2. Model Development

[29] *Jain and Srinivasulu* [2006] used effective rainfall values at time steps t, $t-1$, and $t-2$ (P_t, P_{t-1}, and P_{t-2}) and flow values at time steps $t-1$ and $t-2$ (Q_{t-1} and Q_{t-2}) in order to model the flow at time t (Q_t). Since the focus of the present study is on real-time forecasting, the use of effective rainfall at time t (P_t) was omitted to ensure that only previous values were used to forecast the flow at time t (Q_t). To further test the proposed classification and retraining methods, experiments were also conducted in which flow was forecast three and seven days in advance. In order to evaluate the efficacy of the three model deployment methodologies investigated, five different trials were conducted by dividing the first 13 years of data (1960–1972) into calibration sets of differing lengths (Figure 2). Experiments were performed using differing lengths of calibration data in order to explore the effect that limited calibration data has on an ANN model once deployed in a real-time simulation and to provide a comparison for testing the model deployment approaches. Each of the five calibration data sets (RR-1 to RR-5) were further divided by allocating the first half of the data to the training set and the second half to the testing set. Details of each set are provided in Table 1. The second 13 year period of data (1977–1989) was used as the real-time simulation data in order to test the efficacy of the recalibration approaches (Figure 3). The simulation showed what it would have been like if these models had been deployed over the 13 year period and used in the manner dictated by the retraining method. Even though this was a simulation, it would have been possible to retrain the ANN models in real-time and obtain the forecast. This is because each retrain took of the order of minutes to run, whereas the shortest forecasting horizon was 1 day. This means that there would be plenty of time for retraining and obtaining the forecast if this model was deployed in a real-time application. The performance of all models was also compared to that of a naive forecasting model, which served as a benchmark for the flow forecast at time t ($\widehat{Q}_t$). The naive model is $\widehat{Q}_t = Q_{t-n}$, where n is the forecasting horizon. For example, for a one-step forecast the naïve model sets the forecast for

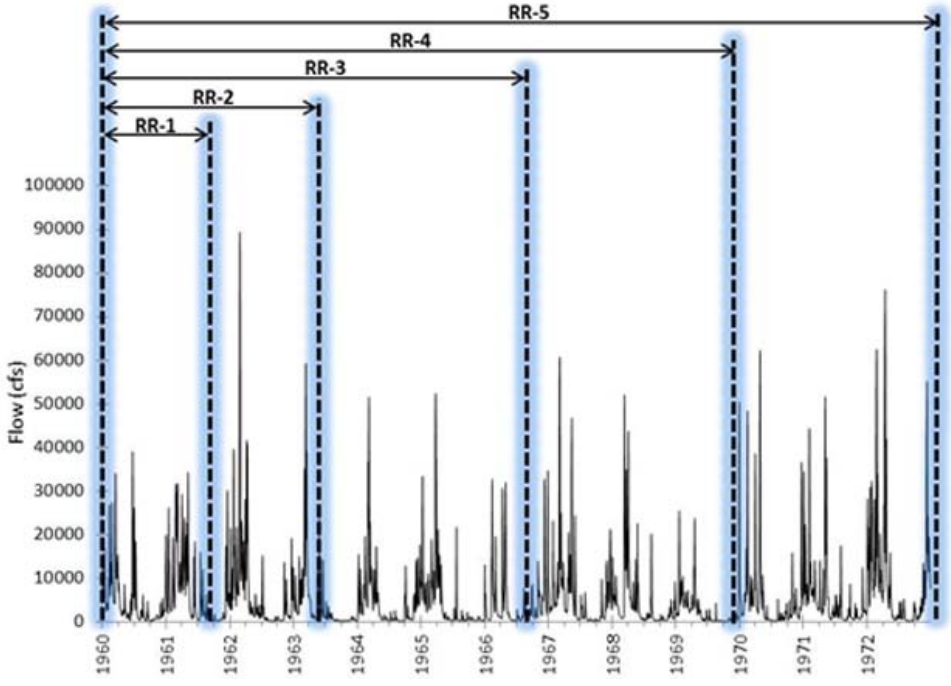

Figure 2. Discretization of the 13 years (1960–1972) of available rainfall-runoff (RR) data into five calibration sets.

Table 1. Train and Test Periods for the Five Rainfall-Runoff (RR) Data Sets

Data Set	Proportion of Total (%)	Train Start Date (dd/mm/yyyy)	Train End Date (dd/mm/yyyy)	No. Train Samples	Test Start Date (dd/mm/yyyy)	Test End Date (dd/mm/yyyy)	No. Test Samples
RR-1	12.5	2/01/1960	24/10/1960	297	25/10/1960	17/08/1961	297
RR-2	25.0	2/01/1960	16/08/1961	593	17/08/1961	1/04/1963	593
RR-3	50.0	2/01/1960	2/04/1963	1187	3/04/1963	2/07/1966	1187
RR-4	75.0	2/01/1960	15/11/1964	1780	16/11/1964	30/09/1969	1780
RR-5	100.0	2/01/1960	2/07/1966	2374	3/07/1966	30/12/1972	2373

flow at time t as the actual flow observed at time $t - 1$. Naive models have zero skill, and therefore provide a way of assessing the skill a particular model offers the forecasting problem.

[30] The type of ANN model used in this study was the feedforward multilayer perceptron (MLP) trained using the backpropagation algorithm. This combination of ANN architecture and training algorithm is by far the most commonly used in water resources applications [see *Maier et al.*, 2010] and provides a good basis upon which to explore different recalibration approaches. *Jain and Srinivasulu* [2006] used MLPs with one hidden layer and applied a trial-and-error approach in order to select the number of hidden nodes in the range of 1 to 20. Since the focus of the current investigation is on understanding the range of applicability of the model and comparing recalibration approaches, the same network geometry was adopted, which consisted of a single hidden layer with five nodes for each model. In contrast with the study performed by *Jain and Srinivasulu* [2006], a linear transfer function was used in the output layer as this is likely to result in superior extrapolation ability. This was confirmed by preliminary trials on the present case study. The trials also indicated that a small learning rate was required for this study. After considering values in the range [0.0001, 0.5], it was found that a learning rate of 0.001 provided the best learning ability. The data were linearly scaled between 0.0 and 1.0. To ensure that overtraining did not occur (i.e., when the network performs well on the training data, but poorly on independent test data), the hold-out method of cross validation was used. This method alternately runs train and test commands, and saves the network with the best test results during the run. After 200 iterations with no further improvement in the test set results, training is stopped.

[31] The ANNs developed using the different calibration sets were used to provide forecasts for the 13 year real-time simulation period. Three deployment approaches were investigated, including: no retraining, always retraining after each new sample, and the proposed method of selectively retraining the model using the SOM-LDE classification method as outlined in section 2. The last approach involves retraining the model each time the SOM-LDE classifier detects an uncharacteristic data pattern. The SOM grid size was determined by the heuristic rule provided by *Vesanto* [1999] as given in equation (3). It should be noted that the SOM grid size grew in proportion to the number of calibration samples as patterns were identified as being uncharacteristic and then added to the calibration data.

3.3. Performance Measures

[32] The performance of all models developed in this study was evaluated using three statistical measures commonly used in water resources modeling. The first measure is the root mean square error (RMSE), as it is suitable as a general measure of model performance and the use of squaring tends to place greater weight on outlier values. Two correlation-based measures were also used, including the coefficient of efficiency (COE), commonly referred to as the Nash-Sutcliffe efficiency and the coefficient of determination (R^2). The COE is the ratio of the MSE to the

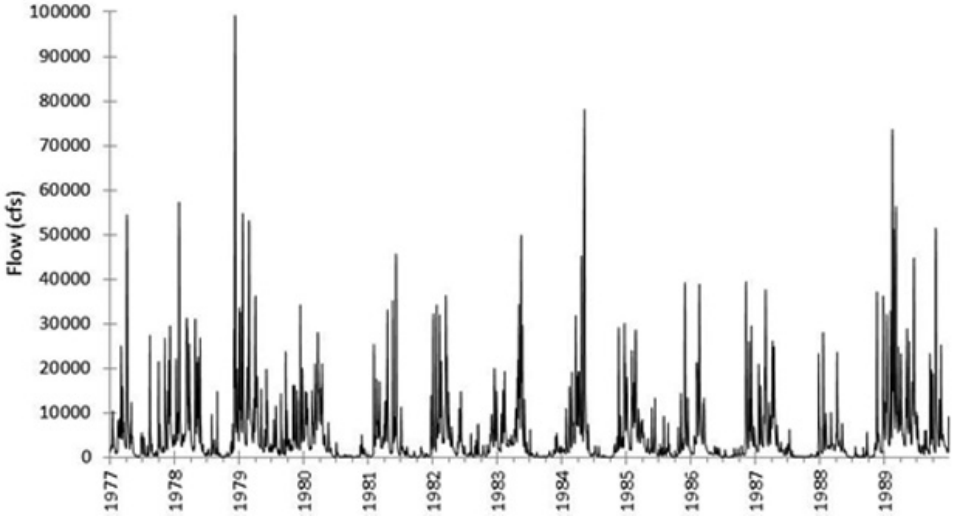

Figure 3. Real-time simulation data spanning 13 years (1977–1989) for the rainfall-runoff case study.

variance of the observed data, subtracted from unity and therefore, measures the ability of a model to predict values which are different from the mean. The coefficient of determination (R^2) is the square of the Pearson product-moment correlation coefficient and measures the proportion of variance in the observed data that is explained by the model [*Legates and McCabe*, 1999]. Both the COE and R^2 are useful because they are measures that are independent of scale and hence, can be used for comparisons between different studies [*Dawson and Wilby*, 2001].

4. Results and Discussion

[33] The real-time simulation period results obtained for the rainfall-runoff case study for the models that were calibrated with data of different lengths and 1 day forecasts are shown in Figure 4. Also shown for comparative purposes is the performance of the naive forecasting model. It can be seen that the MLP model with no retraining (i.e., fixed parameters) was able to outperform the benchmark naive model for all calibration sets, thereby demonstrating its skill relative to the benchmark. A general trend can be

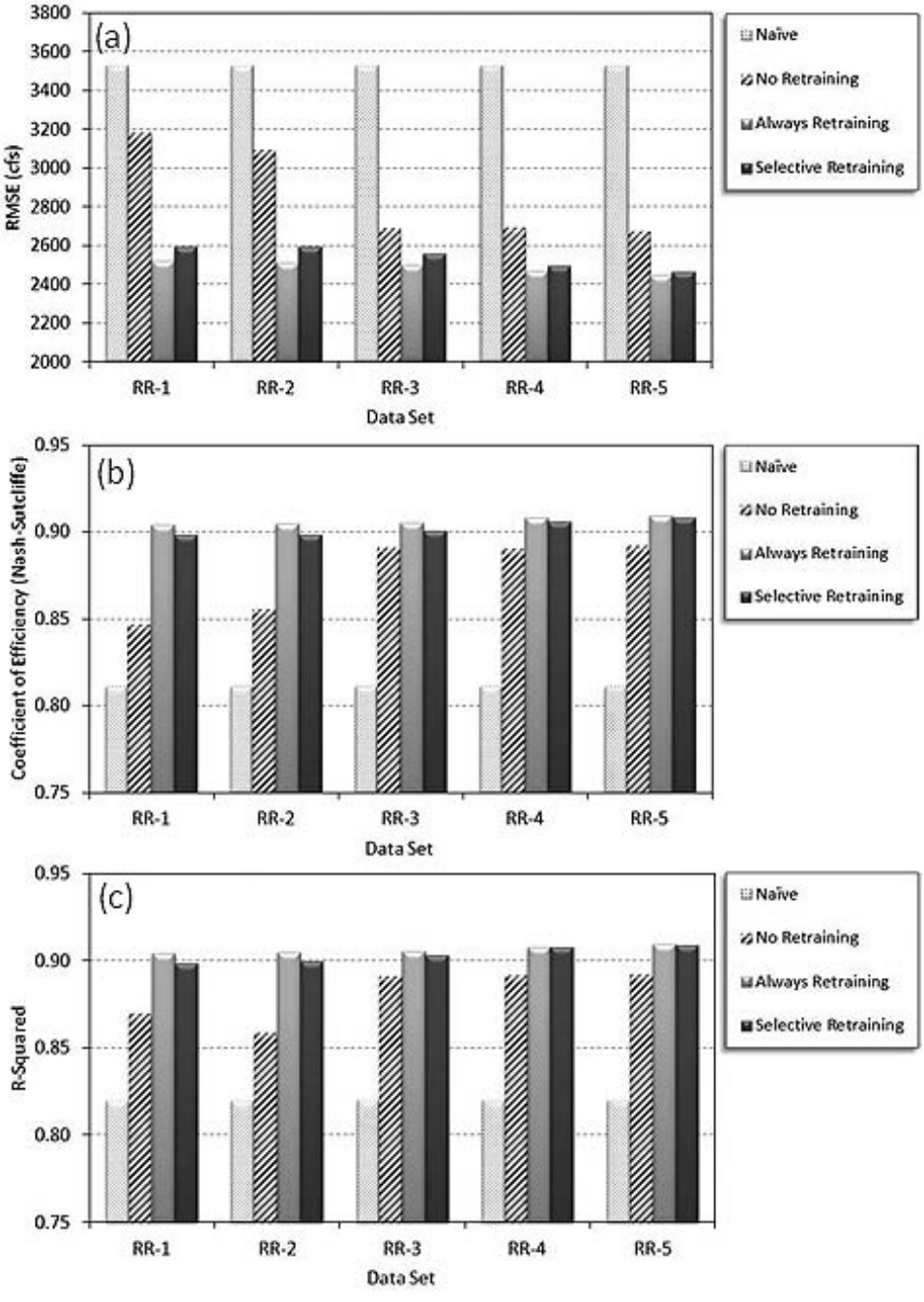

Figure 4. Real-time simulation performance (a) RMSE, (b) coefficient of efficiency and (c) R^2, obtained for the five rainfall-runoff calibration data sets and the retraining methods, forecasting flow 1 day in advance. The naive model forecasts are also shown as a baseline.

observed, whereby performance of the baseline MLP improved as the calibration set increased in size, however, performance plateaued after the RR-3 calibration set. The significantly worse performance on the smaller calibration sets (i.e., RR-1 and RR-2) demonstrates that developing the MLP on fewer data resulted in poor performance when tested on the real-time simulation period since the model was not able to generalize to new unseen patterns. As the calibration data set increased in size, the MLPs' generalization ability improved via a greater diversity of training patterns.

[34] The results obtained when always retraining the MLP and selectively retraining the MLP using the SOM-LDE classifier are also shown in Figure 4. It can be seen that always retraining the model showed significant improvement over no retraining for all calibration sets. The RR-5 data set consists of a 13 year calibration period, and as such, is more likely to contain a greater number of all possible input patterns, whereas RR-1 consists of just over 1.5 years of data and contains a much more limited set of input patterns. Consequently, the level of outperformance generally decreased as the size of the calibration set increased. This is also illustrative of the fact that models developed on smaller calibration sets can benefit more significantly from a periodic recalibration strategy. This is because the model needs to adapt to the new information as it is encountered in order for it to become increasingly robust with time. As the variability of a data set increases, the problem of having a small calibration set becomes of greater concern.

[35] It is also instructive to compare the real-time simulation period results with the performance obtained on the initial calibration data (i.e., the training and testing sets). Table 2 presents the results of the training, testing, and real-time simulation period for the models developed using the RR-1 data set. When calibration data were quite limited, as was the case for the RR-1 data set, the two recalibration strategies were able to achieve real-time simulation period performance metrics that were similar, and even slightly improved, compared to those obtained on the training and testing data sets. In contrast, not retraining the model resulted in significant performance deterioration relative to the initial training and testing results. This highlights that the retraining strategies were successful in maintaining a similar level of error during the real-time simulation period as observed during the initial calibration of the model. This was generally maintained across all five calibration data sets, however, as the amount of initial calibration data increased, the real-time performance of the no retraining strategy became similar to that obtained on the initial training and testing sets. This provided further evidence that recalibration strategies are particularly important in cases where the initial calibration data are limited and not sufficiently representative of future data. Since it cannot generally be known a priori whether the calibration data will be representative of future data, it is better to adopt a recalibration strategy when considering real-time deployment, in order to provide some insurance that model performance will be maintained at an acceptable level.

[36] The selective retraining strategy uses the SOM-LDE classifier to diagnose the uncharacteristic data points, issues a warning for the forecast, and then retrains the model once the corresponding output pattern has been collected. The number of uncharacteristic data points identified by the SOM-LDE classifier varied between approximately 6% and 8% of the total patterns encountered, depending on the initial calibration set used (Table 3). However, even though the range was narrow, a clear trend was observed whereby the amount of uncharacteristic data detected decreased as the size of the calibration set increased. The models developed using the RR-1 and RR-2 calibration sets identified significantly more uncharacteristic data patterns through the early part of the real-time simulation period (approximately the first 3.5 years). As these patterns were incorporated into the model and the calibration set became more representative, the classifier then detected uncharacteristic patterns at a rate that was similar to that for the models developed using the longer calibration sets (i.e., RR-3 to RR-5). By incorporating only the key patterns early on, the number of extra data samples that were required to be added to the RR-1 and RR-2 calibration sets was kept to a minimum. This explains why the number of uncharacteristic data did not vary much between the different models.

[37] Despite only requiring retraining such a small fraction of the time, the selective retraining approach was able to provide similar performance to the approach that retrained the model 100% of the time. For example, when using the MLP model developed with the RR-1 calibration set, the SOM-LDE classifier diagnosed 370 uncharacteristic input patterns. Therefore, 370 warnings were issued with these forecasts, representing 7.8% of the time that retraining was required to be performed to adapt the MLP model to these new patterns. Consequently, there are significant computational benefits to the selective retraining approach, and these benefits become particularly relevant for higher frequency case studies when the time available to perform the model recalibration is constrained. The computational burden of the SOM-LDE diagnostic test presented in this study is small relative to that required to perform a recalibration of the MLP model. However, it is an important consideration that if a diagnostic test is used that requires a significant amount of time to run relative to retraining, then it may be beneficial to simply consider retraining after each new sample is collected. However, in this instance one would lose the ability to issue warnings with the forecast at

Table 2. Training, Testing, and Real-Time Simulation Period Performance for the MLPs Developed Using the RR-1 Data Set and Forecasting Flow 1 Day in Advance

Period	RMSE (cfs)	COE	R^2
Training	2675	0.842	0.849
Testing	2780	0.874	0.891
Real-Time—No Retraining	3181	0.847	0.870
Real-Time—Always Retraining	2521	0.904	0.904
Real-Time—Selective Retraining	2592	0.898	0.898

Table 3. Proportion of Samples Detected as Uncharacteristic for Each Rainfall-Runoff Data Set

Model	RR-1	RR-2	RR-3	RR-4	RR-5
Selective Retraining	7.8%	7.2%	7.0%	6.5%	6.1%

times when the new input sample is detected to be outside of the calibration domain.

[38] The classification performed by the SOM-LDE classifier for the real-time simulation period is presented graphically in Figure 5, which plots the logarithm of the local uncharacteristic factors (LUF) with flow and effective rainfall. The points that were identified as being sufficiently represented by the calibration data are shown as dots with the uncharacteristic data points represented as squares. Also shown is a linear regression line fitted to the uncharacteristic data points. These results are for the MLP model developed using the RR-1 calibration set, which contains comparatively lower flow and rainfall events than the real-time simulation period. As might be expected, the characteristic data points are clustered at these lower values, with significantly higher LUFs being assigned to the higher flow and rainfall input patterns. The clustering between characteristic and uncharacteristic patterns is relatively distinct in this example, which demonstrates the effectiveness of using the SOM-LDE classifier to selectively retrain the model.

[39] Time series plots of the MLP 1 day forecasts obtained using each model deployment strategy are shown in Figure 6.

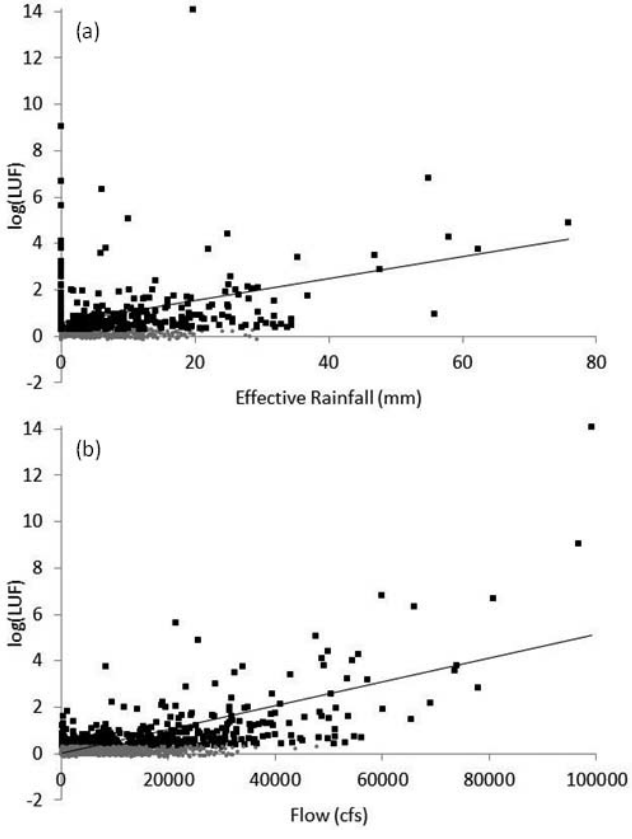

Figure 5. Local uncharacteristic factors (LUF) for the rainfall-runoff real-time simulation period (a) effective rainfall, and (b) flow. Gray shaded circles denote characteristic data vectors and black filled squares denote uncharacteristic vectors. Results are for the MLP model developed using the RR-1 calibration set and forecasting flow 1 day in advance.

These results display a 2 year section of the real-time simulation (1978–1979) that contained the peak flow event of 99,100 cubic feet per second (cfs). It is apparent that without retraining, the MLP model produces poor forecasts of the larger flow events (Figure 6a), because these events were clearly outside of the domain of the calibration data and required the model to extrapolate. Always retraining the MLP model provided a significant improvement in the ability to forecast the larger flow events (Figure 6b), which shows that although the model was developed on a limited set of calibration data, the retraining strategy effectively allowed the model to adapt to these new patterns. Selective retraining also greatly improved the MLP's ability to forecast the larger flow events, although the peak flow was over-predicted in this case (Figure 6c). The general improvement in forecasting the peak flow events was evident despite the fact that the SOM-LDE classifier was very judicious in identifying when retraining was necessary. The same pattern was also evident across the entire real-time validation period, as can be seen in Figure 7, which presents scatterplots of the 1 day forecasts for the entire real-time simulation period under the three retraining approaches. It is important to note here that the forecast for the first new event of high flow is not good despite the retraining strategies, as this pattern is only included in the calibration data after it had occurred (i.e., after the poor forecast). However, the inclusion of this event then has a positive impact on subsequent forecasts as this type of event is then incorporated into the calibration domain. For brevity, only the results for calibration data sets RR-1, RR-3, and RR-5 are presented, as they are indicative of the general trend that was observed. As expected, the level of outperformance of the retraining strategies, compared with the no retraining approach, diminished as the amount of calibration data increased. The degree of variability and the range of the values included in the calibration data are of importance here. In this case study, the data are highly variable and the first few years of data contain lower flows, such that by increasing the length of the calibration data, more extreme patterns are included and the resulting forecasts on the real-time simulation period were then improved.

[40] The real-time simulation results for each deployment method for the 3 and 7 day ahead forecasting periods are shown in Figures 8 and 9, respectively. The purpose of this investigation was not to derive an optimal model in these cases, but rather to deploy the models under the same conditions and with the same inputs as used earlier but with a longer forecast period. Despite the fact that model performance was considerably degraded relative to the 1 day forecasts, as expected, the general trends observed were very similar. For both the 3 and 7 day forecasts, the MLP with no retraining (i.e., fixed weights) provided significant improvement relative to the naïve model, thereby demonstrating its skill relative to the benchmark. As expected, the baseline MLP showed improvement in performance as the size of the calibration data was increased for both the 3 and 7 day forecasts. The active retraining strategies also resulted in a significant improvement in performance compared with the baseline MLP and the level of outperformance decreased as the size of the calibration data increased. The results obtained when using longer forecast horizons demonstrate that even when the forecasts contain considerable error, there is still significant improvement to be made

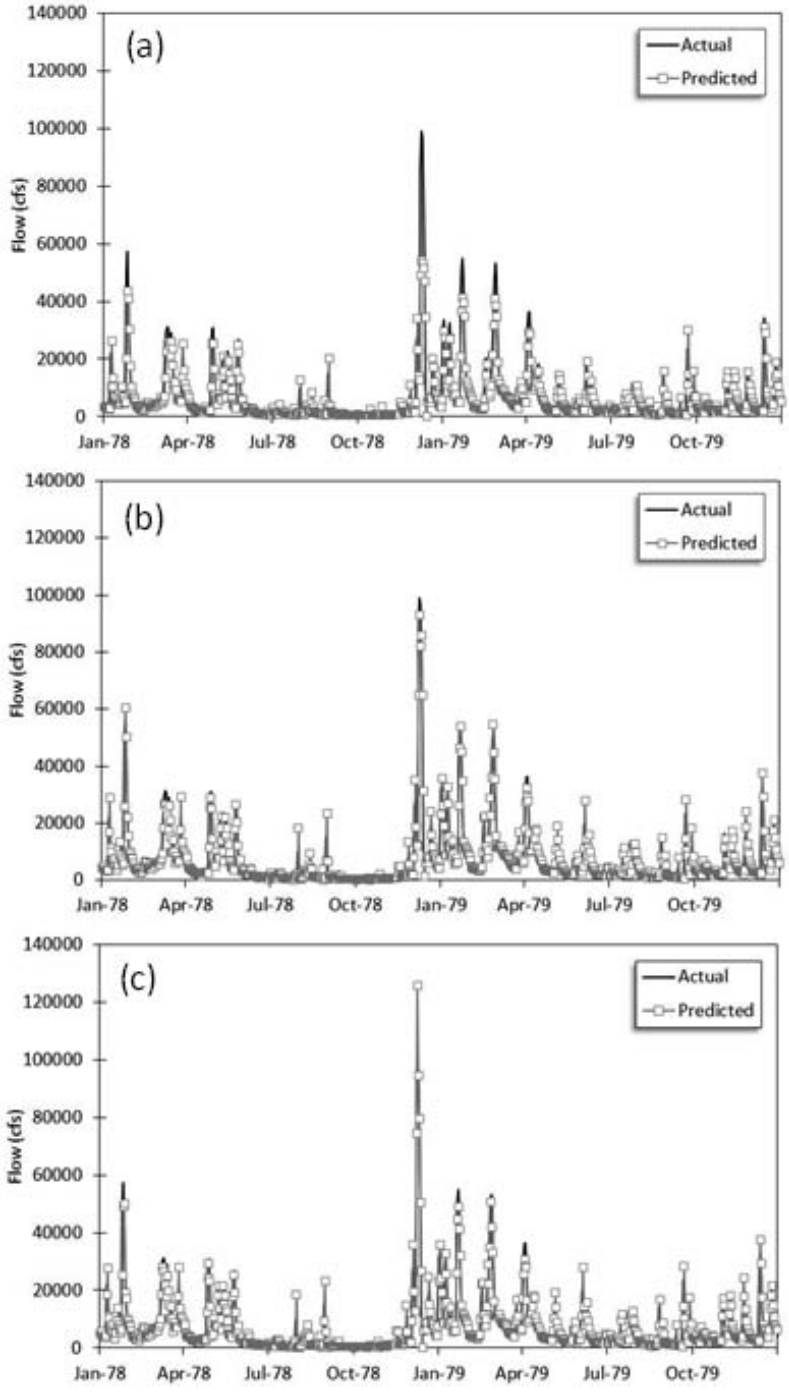

Figure 6. Real-time simulation 1 day ahead flow forecasts for 1978–1979 using RR-l for calibration and (a) no retraining, (b) always retraining, and (c) selective retraining as the deployment approach.

by utilizing an active retraining deployment strategy such as that presented in this paper.

5. Summary and Conclusions

[41] When an ANN model is deployed in an operational environment, there is a need to determine if new input patterns are representative of the data used in calibrating the model. In this paper, a novel detection system based on a SOM-LDE classifier is presented for identifying uncharacteristic data patterns. Once an uncharacteristic pattern is detected, it is added to the calibration data set and the model is retrained after the relevant output variable has been collected. In so doing, the model is able to cater to nonstationarity in the data and effectively adapt to new information as it is encountered.

[42] Selectively retraining the model using the SOM-LDE classifier was compared with the simpler approaches

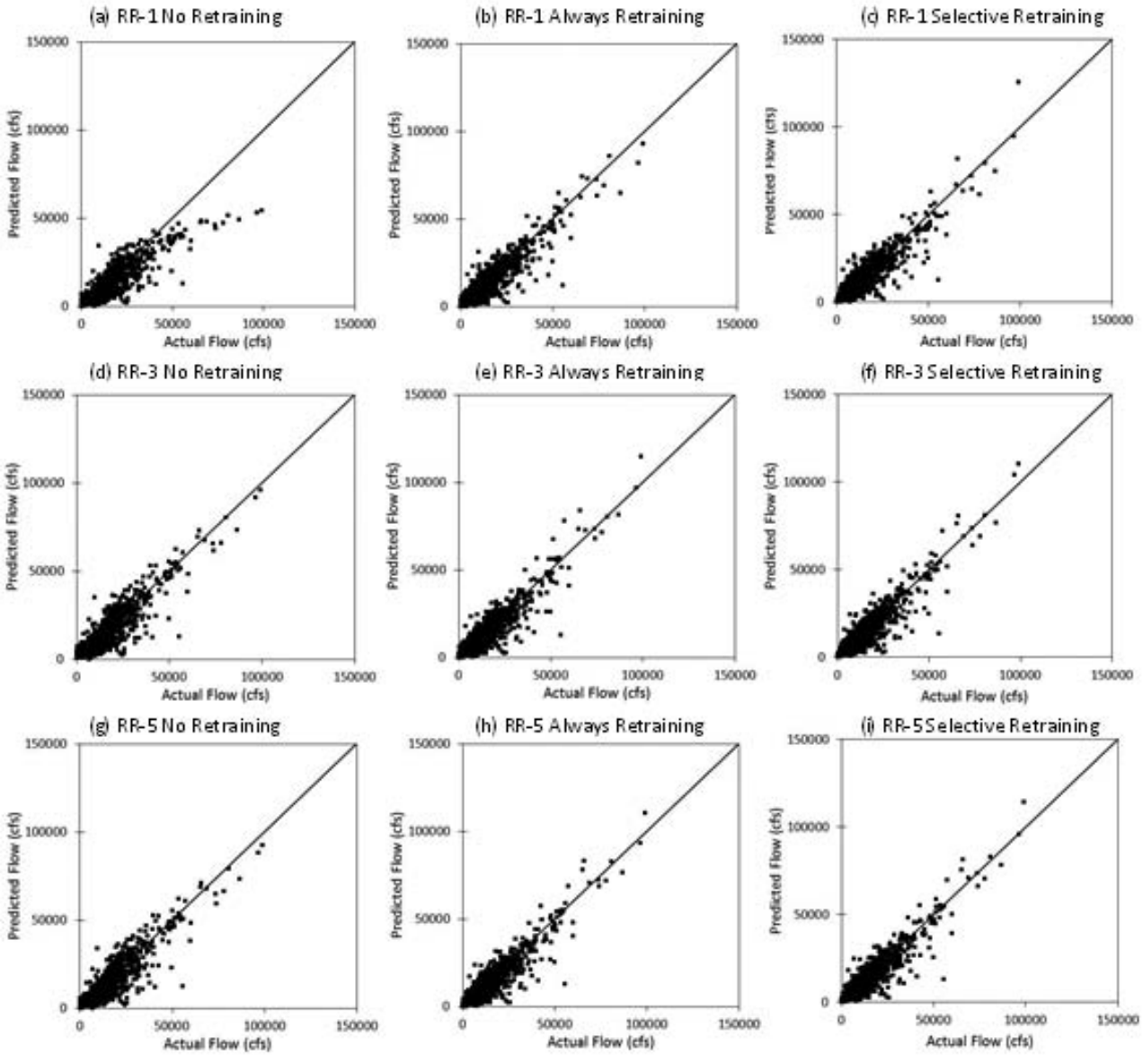

Figure 7. Scatterplots of the real-time simulation period 1 day ahead forecasts of flow versus actual flow for data sets RR-1, RR-3, and RR-5 using the three model deployment approaches.

of no retraining (i.e., keeping the model parameters fixed) and always retraining the model using a real-world case study. All three model deployment approaches were applied to a 13 year rainfall-runoff forecasting real-time simulation, where forecasts were made 1 day in advance. The results from this investigation provided firm evidence that periodically retraining the model resulted in significant improvements when compared to keeping the model parameters fixed. By evaluating models developed using differing amounts of calibration data, it was also determined that the degree of outperformance achieved by actively retraining the model was greatest when only limited data were available in the model development process. In highly variable data sets such as the rainfall-runoff study investigated here, the problem of limited model development data is that it becomes increasingly unlikely that the calibration data contain a sufficiently representative range of patterns likely to be encountered in the future. This was clearly observed in the rainfall-runoff case study as the models developed using smaller calibration sets significantly underperformed compared with those developed with larger calibration sets and failed to accurately forecast the high flow events.

[43] The results also indicated that the SOM-LDE classifier provided an effective means of detecting when retraining is required. For the case study investigated, the amount of uncharacteristic data detected by the classifier was generally between 6% and 8%, and despite only requiring the model to be retrained such a small fraction of the time, similar model performance to the approach that retrained 100% of the time was achieved. Therefore, the selective retraining approach was also more computationally efficient than the approach of always retraining the model. The added

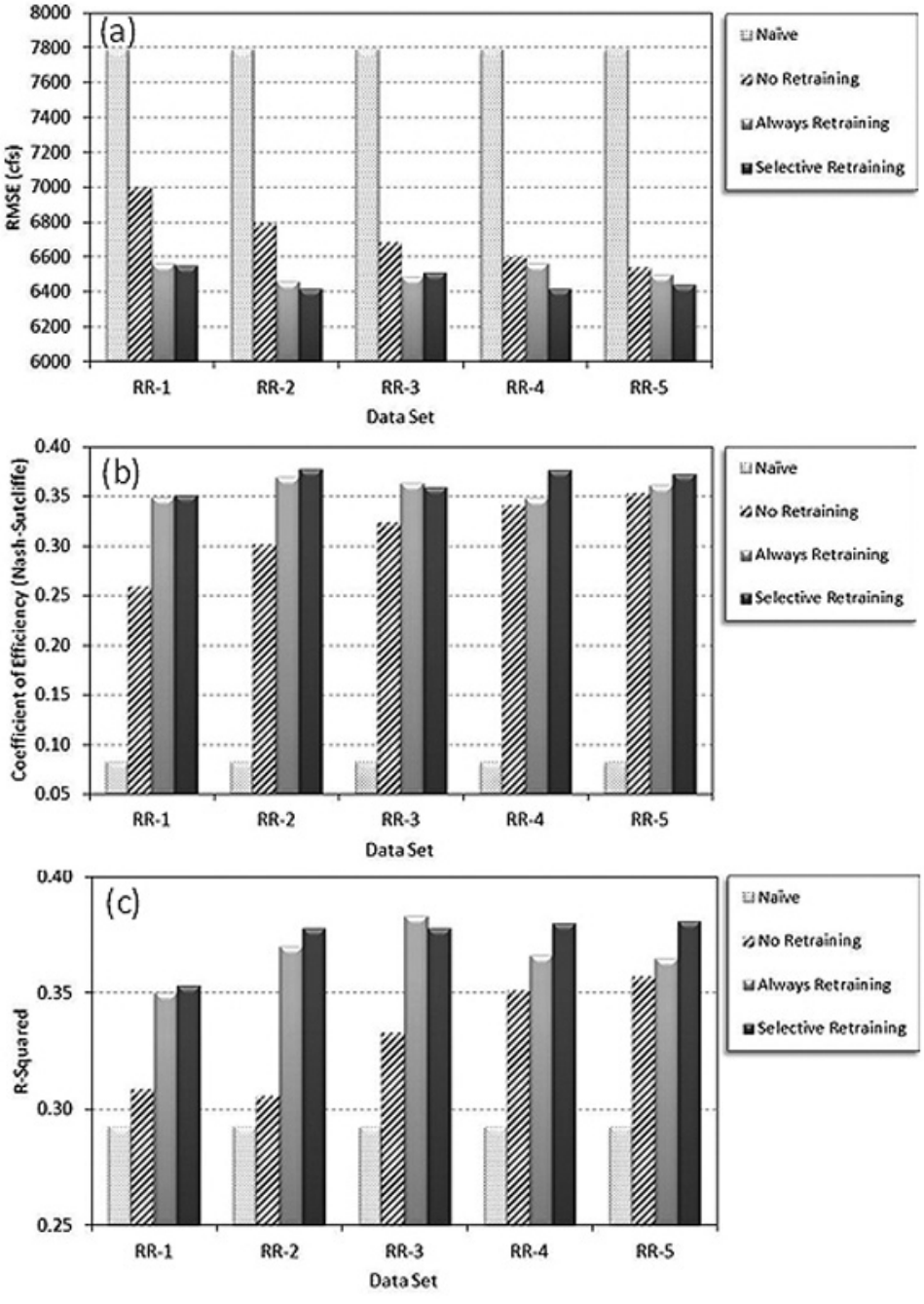

Figure 8. Real-time simulation performance (a) RMSE, (b) coefficient of efficiency and (c) R^2, obtained for the five rainfall-runoff calibration data sets and the model deployment approaches, forecasting flow 3 days in advance. The naive model forecasts are also shown as a baseline.

advantage of using the SOM-LDE classifier is that it is able to provide a warning when there is likely to be considerable uncertainty in the model's forecast due to an uncharacteristic input sample being detected. In this study the classifier was used to determine when to selectively retrain, however, there is no reason why the classifier could not also be used in conjunction with any model deployment scenario, in order to provide warnings of when the forecast is likely to contain a larger degree of uncertainty.

[44] The deployment approaches were also investigated when using longer forecasting horizons, specifically 3 and 7 days. The results were broadly in line with those obtained for the 1 day forecasts. The baseline MLP's performance improved as the amount of calibration data was increased. The retraining approaches provided significant improvement in performance relative to no retraining.

[45] Overall, the investigation highlights the necessity for periodically recalibrating ANN models when they are deployed in a real-time operational setting. The proposed SOM-LDE classifier provided an effective means for partitioning the calibration data and estimating the local density of new input patterns relative to the calibration points.

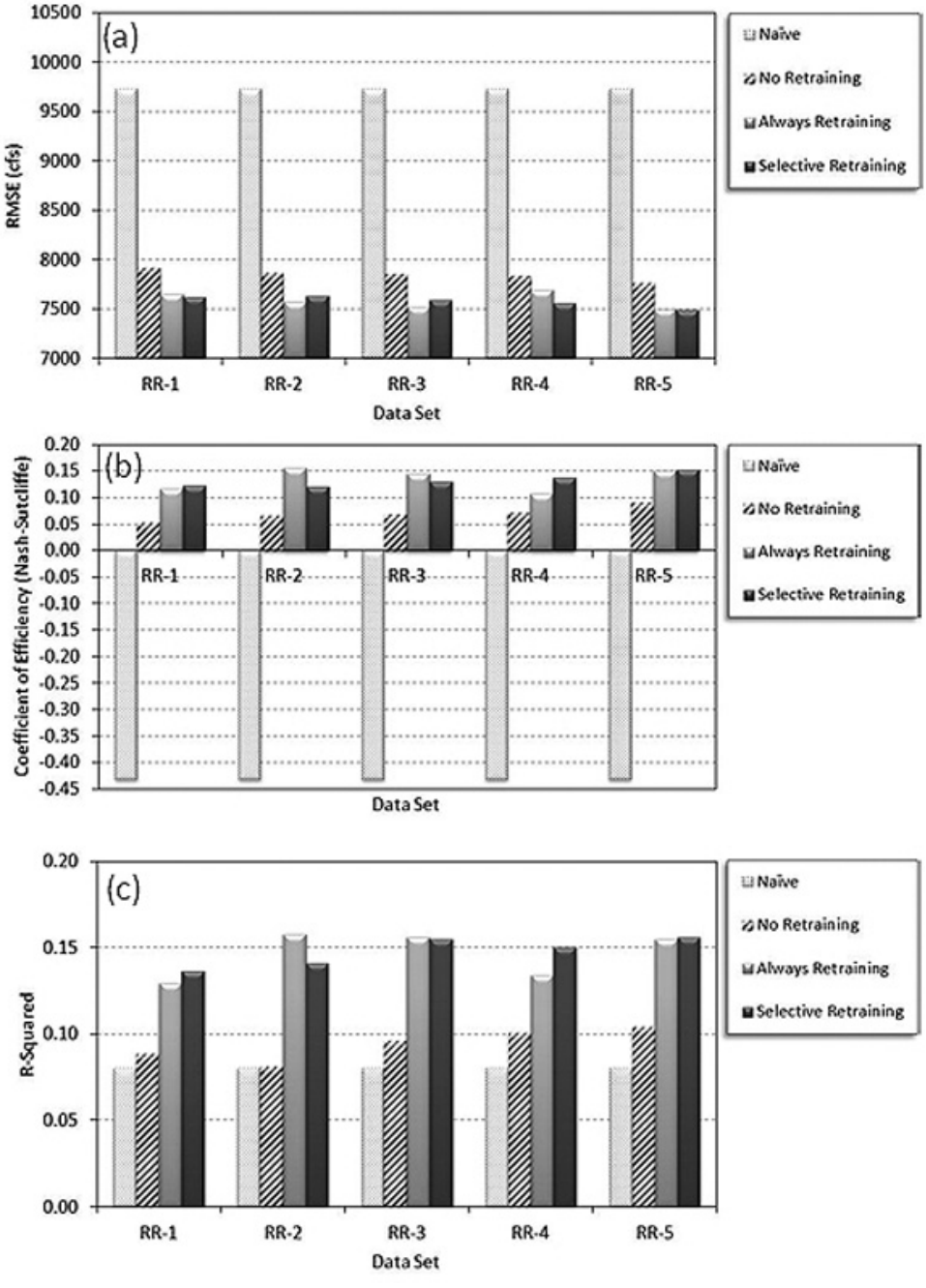

Figure 9. Real-time simulation performance (a) RMSE, (b) coefficient of efficiency and (c) R^2, obtained for the five rainfall-runoff calibration data sets and the model deployment approaches, forecasting flow 7 days in advance. The naive model forecasts are also shown as a baseline.

In so doing, it is possible to define the range of applicability of ANN models. Future research should be directed toward additional testing of these methods on further case studies and toward improving approaches for detecting uncharacteristic data patterns. In applications where very high-frequency forecasts are required, there may be little time available between subsequent forecasts. In such cases, it would be useful to investigate more efficient methods for incorporating uncharacteristic data patterns such as online (incremental) weight updates. The SOM-LDE classifier and model recalibration methodology developed in this research has wider applicability than just ANN models. In fact, the approach could be used for a wide range of other empirical or data mining based prediction methods. Further studies investigating how the methodology performs when deploying other data-driven prediction models would be a good avenue of future research.

[46] **Acknowledgment.** The authors would like to gratefully acknowledge Professor Ashu Jain from the Department of Civil Engineering, Indian Institute of Technology Kanpur, for kindly providing the rainfall-runoff data set used in this study.

References

Abrahart, R. J., F. Anctil, P. Coulibaly, C. W. Dawson, N. J. Mount, L. M. See, A. Y. Shamseldin, D. P. Solomatine, E. Toth, and R. L. Wilby (2012), Two decades of anarchy? Emerging themes and outstanding challenges for neural network river forecasting, *Progr. Phys. Geogr.*, *36*(4), 480–513, doi:10.1177/0309133312444943.

Adeloye, A. J., R. Rustum, and I. D. Kariyama (2011), Kohonen self-organizing map estimator for the reference crop evapotranspiration, *Water Resour. Res.*, *47*(8), W08523, doi:10.1029/2011WR010690.

Aqil, M., I. Kita, A. Yano, and S. Nishiyama (2007), Neural networks for real time catchment flow modeling and prediction, *Water Resour. Manage.*, *21*(10), 1781–1796, doi:10.1007/s11269-006-9127-y.

ASCE Task Committee on Application of Artificial Neural Networks in Hydrology (2000), Artificial neural networks in hydrology. I: Preliminary concepts, *J. Hydrol. Eng.*, *5*(2), 115–123.

Atkinson, A. C. (1994), Fast very robust methods for detection of multiple outliers, *J. Am. Stat. Assoc.*, *89*, 1329–1339.

Bowden, G. J., H. R. Maier, and G. C. Dandy (2002), Optimal division of data for neural network models in water resources applications, *Water Resour. Res.*, *38*(2), 1010, doi:10.1029/2001WR000266.

Bowden, G. J., G. C. Dandy, and H. R. Maier (2005a), Input determination for neural network models in water resources applications. Part 1—Background and methodology, *J. Hydrol.*, *301*(1–4), 75–92.

Bowden, G. J., H. R. Maier, and G. C. Dandy (2005b), Input determination for neural network models in water resources applications. Part 2. Case study: Forecasting salinity in a river, *J. Hydrol.*, *301*(1–4), 93–107.

Brath, A., A. Montanari, and E. Toth (2002), Neural networks and non-parametric methods for improving real-time flood forecasting through conceptual hydrological models, *Hydrol. Earth Syst. Sci.*, *6*(4), 627–639.

Breunig, M. M. (2000), LOF: Identifying density-based local outliers, in *ACM SIGMOD 2000 Int. Conf. on Management of Data*, Dallas, TX, Assoc. for Computing Machinery, New York.

Bruen, M., and J. Q. Yang (2005), Functional networks in real-time flood forecasting—A novel application, *Adv. Water Resour.*, *28*(9), 899–909, doi:10.1016/j.advwatres.2005.03.001.

Cacoullous, T. (1966), Estimation of a multivariate density, *Ann. Inst. Stat. Math. (Tokyo)*, *18*(2), 179–189.

Chang, F. J., L. C. Chang, and H. L. Huang (2002), Real-time recurrent learning neural network for stream-flow forecasting, *Hydrol. Processes*, *16*(13), 2577–2588, doi:10.1002/hyp.1015.

Chen, S. T., and P. S. Yu (2007), Real-time probabilistic forecasting of flood stages, *J. Hydrol.*, *340*(1–2), 63–77, doi:10.1016/j.jhydrol.2007.04.008.

Cigizoglu, H. K. (2003), Estimation, forecasting and extrapolation of river flows by artificial neural networks, *Hydrol. Sci. J.*, *48*(3), 349–361.

Coulibaly, P., F. Anctil, and B. Bobee (2000), Daily reservoir inflow forecasting using artificial neural networks with stopped training approach, *J. Hydrol.*, *230*(3–4), 244–257.

Coulibaly, P., F. Anctil, and B. Bobee (2001), Multivariate reservoir inflow forecasting using temporal neural networks, *J. Hydrol. Eng.*, *6*(5), 367–376.

Dawson, C. W., and R. L. Wilby (2001), Hydrological modelling using artificial neural networks, *Progr. Phys. Geogr.*, *25*(1), 80–108.

Fassnacht, S. R., and J. E. Derry (2010), Defining similar regions of snow in the Colorado River Basin using self-organizing maps, *Water Resour. Res.*, *46*, W04507, doi:10.1029/2009WR007835.

Flood, I., and N. Kartam (1994), Neural networks in civil engineering. I: Principles and understanding, *J. Comput. Civil Eng.*, *8*(2), 131–148.

Giustolisi, O., and D. Laucelli (2005), Improving generalization of artificial neural networks in rainfall-runoff modelling, *Hydrol. Sci. J.*, *50*(3), 439–457.

Goswami, M., and K. M. O'Connor (2007), Real-time flow forecasting in the absence of quantitative precipitation forecasts: A multi-model approach, *J. Hydrol.*, *334*(1–2), 125–140, doi:10.1016/j.jhydrol.2006.10.002.

Goswami, M., K. M. O'Connor, K. P. Bhattarai, and A. Y. Shamseldin (2005), Assessing the performance of eight real-time updating models and procedures for the Brosna River, *Hydrol. Earth Syst. Sci.*, *9*(4), 394–411.

Hawkins, D. (1980), *Identification of Outliers*, 188 pp., Chapman and Hall, Reading, London.

Hawkins, S., H. He, G. Williams, and R. Baxter (2002), Outlier detection using replicator neural networks, in *Data Warehousing and Knowledge Discovery*, edited by Y. Kambayashi, W. Winiwarter, and M. Arikawa, pp. 113–123, Springer, Berlin.

Hu, T. S., K. C. Lam, and S. T. Ng (2005), A modified neural network for improving river flow prediction, *Hydrol. Sci. J.*, *50*(2), 299–318.

Imrie, C. E., S. Durucan, and A. Korre (2000), River flow prediction using artificial neural networks: Generalisation beyond the calibration range, *J. Hydrol.*, *233*(1–4), 138–153.

Jain, A., and S. Srinivasulu (2006), Integrated approach to model decomposed flow hydrograph using artificial neural network and conceptual techniques, *J. Hydrol.*, *317*(3–4), 291–306.

Jain, A., K. P. Sudheer, and S. Srinivasulu (2004), Identification of physical processes inherent in artificial neural network rainfall runoff models, *Hydrol. Processes*, *18*(3), 571–581.

Kingston, G. B., M. F. Lambert, and H. R. Maier (2005), Bayesian training of artificial neural networks used for water resources modeling, *Water Resour. Res.*, *41*(12), W12409, doi:10.1029/2005WR004152.

Knorr, E. M., and R. T. Ng (1998), Algorithms for mining distance-based outliers in large datasets, in *24th VLDB Conference*, pp. 392–403, Morgan Kaufmann, New York.

Kohonen, T. (1982), Self-organized formation of topologically correct feature maps, *Bio. Cyber.*, *43*, 59–69.

Latecki, L., A. Lazarevic, and D. Pokrajac (2007), Outlier detection with kernel density functions, in *Machine Learning and Data Mining in Pattern Recognition*, edited by P. Perner, pp. 61–75, Springer, Berlin.

Legates, D. R., and G. J. McCabe (1999), Evaluating the use of "goodness-of-fit" measures in hydrologic and hydroclimatic model validation, *Water Resour. Res.*, *35*(1), 233–241.

Maier, H. R., and G. C. Dandy (2000), Neural networks for the prediction and forecasting of water resources variables: A review of modelling issues and applications, *Environ. Model. Software*, *15*, 101–124.

Maier, H. R., A. Jain, G. C. Dandy, and K. P. Sudheer (2010), Methods used for the development of neural networks for the prediction of water resource variables in river systems: Current status and future directions, *Environ. Model. Software*, *25*(8), 891–909.

May, R. J., H. R. Maier, and G. C. Dandy (2010), Data splitting for artificial neural networks using SOM-based stratified sampling, *Neural Networks*, *23*(2), 283–294.

Minns, A. W., and M. J. Hall (1996), Artificial neural networks as rainfall-runoff models, *Hydrol. Sci. J.*, *41*(3), 399–417.

Nag, A., A. Mitra, and S. Mitra (2005), Multiple outlier detection in multivariate data using self-organizing maps title, *Comput. Stat.*, *20*(2), 245–264.

Napolitano, G., L. See, B. Calvo, F. Savi, and A. Heppenstall (2010), A conceptual and neural network model for real-time flood forecasting of the Tiber River in Rome, *Phys. Chem. Earth*, *35*, 187–194.

Parasuraman, K., A. Elshorbagy, and S. K. Carey (2006), Spiking modular neural networks: A neural network modeling approach for hydrological processes, *Water Resour. Res.*, *42*(5), W05412, doi:10.1029/2005WR004317.

Parzen, E. (1962), On estimation of probability density function and mode, *Ann. Math. Stat.*, *33*, 1065–1076.

Pearce, A. R., D. M. Rizzo, and P. J. Mouser (2011), Subsurface characterization of groundwater contaminated by landfill leachate using microbial community profile data and a nonparametric decision-making process, *Water Resour. Res.*, *47*(6), W06511, doi:10.1029/2010WR009992.

Petrovskiy, M. I. (2003), Outlier detection algorithms in data mining systems, *Program. Comput. Software*, *29*(4), 228–237.

Sahoo, G. B., and C. Ray (2008), Microgenetic algorithms and artificial neural networks to assess minimum data requirements for prediction of pesticide concentrations in shallow groundwater on a regional scale, *Water Resour. Res.*, *44*(5), W05414, doi:10.1029/2007WR005875.

Scott, D. (1992), *Multivariate Density Estimation: Theory, Practice, and Visualization (Wiley Series in Probability and Statistics)*, Wiley, New York.

Sharma, A. (2000), Seasonal to interannual rainfall probabilistic forecasts for improved water supply management: Part 1—A strategy for system predictor identification, *J. Hydrol.*, *239*(1–4), 232–239.

Tax, D. M. J., and P. W. Duin (1998), Outlier detection using classifier instability, in *Advances in Pattern Recognition*, edited by A. Amin, et al., pp. 593–601, Springer, Sydney.

Tokar, A. S., and P. A. Johnson (1999), Rainfall-runoff modeling using artificial neural networks, *J. Hydrol. Eng.*, *4*(3), 232–239.

Vesanto, J. (1999), SOM-based data visualization methods, *Intel. Data Anal.*, *3*(2), 111–126.

Xiong, L. H., and K. M. O'Connor (2002), Comparison of four updating models for real-time river flow forecasting, *Hydrol. Sci. J.*, *47*(4), 621–639.