J Inno Sci Eng, 2025, 9(1):62-77

DOI:https://doi.org/ 10.38088/jise.1375510

Research Article

Comparison of Levenberg-Marquardt and Bayesian Regularization Learning Algorithms for Daily Runoff Forecasting

Aslı Bor [1,2*], Merve Okan[2]

[1] *Norwegian University of Science and Technology, Department of Civil and Environmental Engineering, N-7491, Trondheim, Norway*

[2] *İzmir University of Economics, Department of Civil Engineering, 35330, İzmir, Türkiye*

ARTICLE INFO

Received Date: 14/10/2023
Accepted Date: 17/01/2025

Cite this paper as:
Bor, A., & Okan, M. (2025). Comparison of Levenberg-Marquardt and Bayesian Regularization Learning Algorithms for Daily Runoff Forecasting. *Journal of Innovative Science and Engineering.* 9(1), 62-77.

*Corresponding author: Aslı Bor
E-mail:asli.turkben@ieu.edu.tr

Keywords:
Discharge forecasting
Rainfall-runoff process
Artificial neural network
Euphrates-Tigris basin

ABSTRACT

In this study, Multilayer Perceptron (MLP) with Levenberg-Marquardt and Bayesian Regularization algorithms machine learning methods are compared for modeling of the rainfall-runoff process. For this purpose, daily flows were forecast using 5844 discharge data monitored between 1999 and 2015 of D21A001 Kırkgöze gauging station on the Karasu River operated by DSI. 6 scenarios were developed during the studies. Our findings indicate that the estimated capability of the Bayesian Regularization algorithm were close to with Levenberg-Marquardt algorithm for training and testing, respectively. This study shows that different network structures and data representing land features can improve prediction for longer lead times. We consider that the ANN model accurately depicted the Karasu flows, and that our study will serve as a guide for more research on flooding and water storage.

1. Introduction

In order to estimate the rainfall-runoff response of catchments and forecast hydrological droughts and flood events that result in fatalities and financial loss, hydrological models are essential applications [1]. To determine water capacities, countries establish a sparse hydrometric data collection network on the surface of rivers. However, estimating water capacity is not an easy work because of the complexity of physical parameters affecting stream flows. This complicated system, and the limits to existing hydrological information create significant uncertainty. The accuracy and capability of flow estimation models may have a direct effect on decisions related to water resources management. That's why; new estimation methods can be

investigated to improve the existing ones. Machine learning is a term used to refer to the area of artificial intelligence that is data-based and contains traits that enable self-adaptability. In recent years, artificial neural network (ANN), is a widespread artificial intelligence method while solving some problems of hydraulic and water resource engineering. While physical models are expensive to build and need elaborate input data, a data-driven model such as ANN is simpler to use and depends just on the availability of climate data [2]. The study findings display that ANN can more accurately forecast when it is compared with both the traditional regression techniques and the current physical-based models, using a wider range of conditions [3-21]. Within the context of hydrological forecasting, the latest experiments demonstrate that ANNs can be a promising alternative for simulating the flow. ANN captures the behaviour of a system by using a training algorithm that minimizes the error function when finding the most suitable connection weights.

The principal objective of the study is to compare the performance of the artificial intelligence techniques, Multilayer Perceptron (MLP) with training algorithms of Levenberg-Marquardt (LM) and Bayesian Regularization (BR) in the forecasting of daily flows. For this purpose, D21A001 Kırkgöze gauging station on the Karasu River, a branch of the Euphrates River, one of the major water sources of Turkey, was selected for case study. Daily runoff forecasting for Euphrates-Tigris Basin is worth to be studied due to encountered frequent flood and drought times in the basin at the past. 6 different scenarios were developed for the forecasts and the optimal scenario was identified.

2. Methodology

2.1. Study site

The Euphrates-Tigris basin covers approximately 127304 km^2 area and it has 1009.87 m height. The average rainfall in the Euphrates-Tigris Basin is 540.1 mm $year^{-1}$, and the average annual flow is 31.61 km^3 which makes it the largest basin in Turkey in terms of average annual flow rate. There have been frequent flood and drought times in the basin since ancient times, causing serious damage to the country's economy. Hence, the estimation of the stream flow in the Euphrates-Tigris basin is particularly important in terms of the effective operation of water resources systems and the reduction of flood damage. Daily rainfall and runoff (discharge) data set is used at D21A001 Kırkgöze gauging station on the Karasu River (Figure 1), a branch of the Euphrates River, for 16 years from 1 October 1999 to 30 September 2015 (5844 data).

The daily runoff data was obtained from DSI flow observation annuals, and temperature and precipitation data was obtained from NASA POWER Data Access Viewer. Information about D21A001 Kırkgöze gauging is given in Table 1.

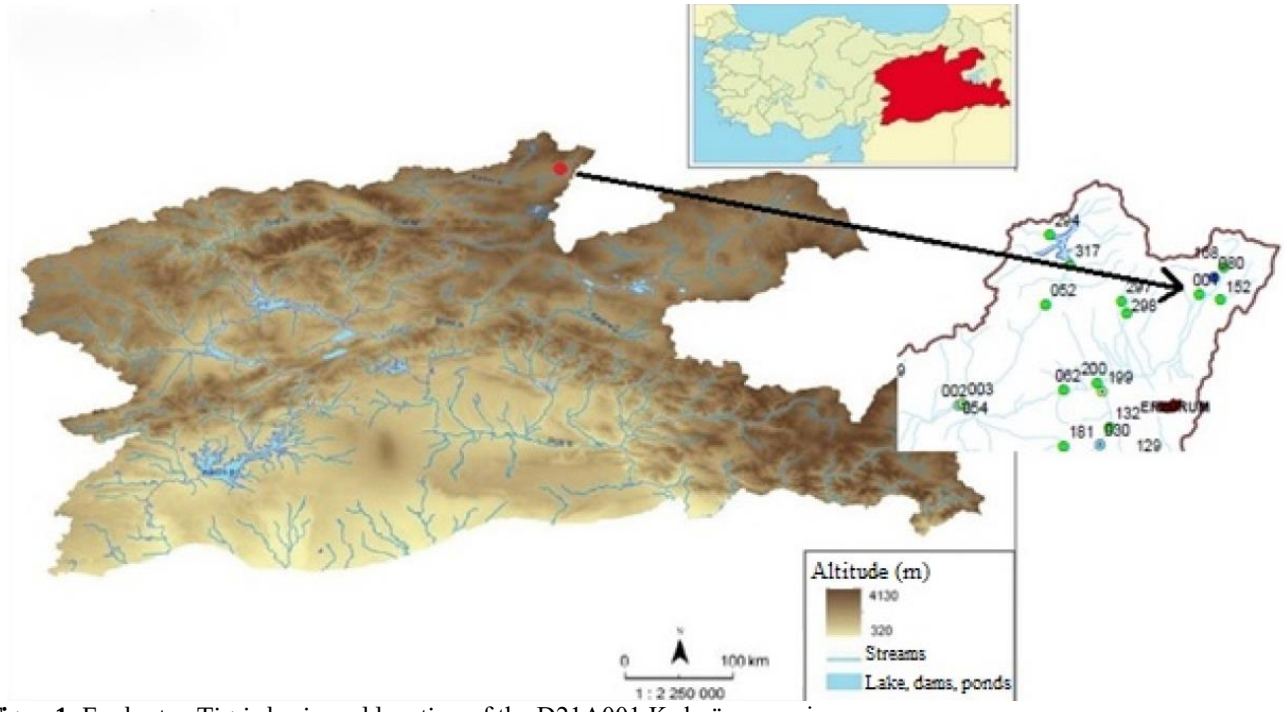

Figure 1: Euphrates-Tigris basin and location of the D21A001 Kırkgöze gauging.

Table 1: Information about Karasu River Kırkgöze gauging.

Station Number:	D21A001
Stream:	Karasu
Station:	Kırkgöze
Management:	DSİ
Altitude (m):	1830
Drainage Area (km²):	232.2
Observation Period:	1961-2016
Latitude:	40°6'29" N
Longitude:	41°23'8" E

2.2. Artificial Neural Networks

One of the most commonly known artificial intelligence techniques in the discipline of water resources engineering is the artificial neural network (ANN). Learning, association, classification, generalization, feature determination, and optimization are just a few applications of artificial neural networks (ANN), which are implemented to nonlinear and mathematical modelling problems [22]. ANN was inspired by the working principle of biological neural networks, which are composed of synapses, axons, dendrites, and nuclei in the brain. In the field of hydrology, a multilayer perceptron is one of the most prevalent network structures (MLP). MLP is used for problems such as classification, prediction, recognition, interpretation, and identification. In recent years, researchers have studied the capabilities of Multilayer Perceptron (MLP) models for the estimation of river flow [11], [13], [23]–[25]. MLP is composed of 3 principal layers as the input, hidden and output layer (Figure 2). MLP can have multiple hidden layers. Haykin (1998) [26] explains MLP more elaborately.

ANN itself generates output data against the input data, that is, it trains the examples given, and then aims to predict the desired data according to its generalization. It is not certain how many hidden neurons there should be in an ANN, the number may vary depending on the problem, data and number of variables. Therefore, when estimating with an ANN model, different hidden neuron numbers should be tested to find the network structure that can optimize the estimation.

The main disadvantage of ANN is that it may not obtain optimum estimates at the first time because different ANN structures are created for different hidden neuron numbers and weight coefficient values, and each ANN makes a different estimate. It is necessary to the method of use trial and to determine which network can optimally predict from the created networks. This does not guarantee that the solution found is the best solution; in other words

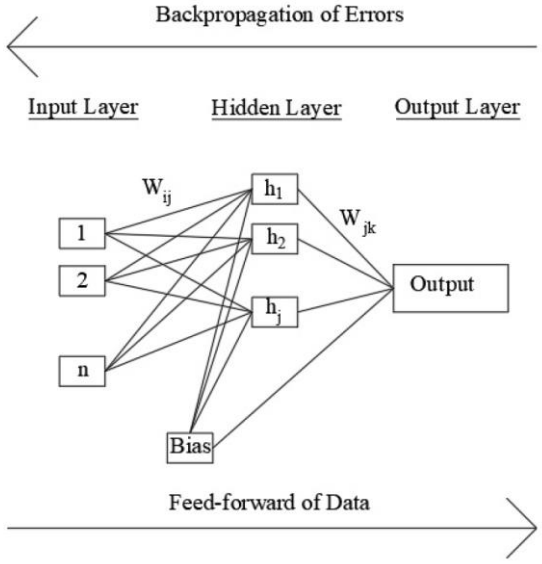

Figure 2: Architecture of multi-layer ANN.

ANN can produce acceptable solutions without guaranteeing these are the best solutions [27].
MLP is trained according to the instructional learning strategy. This approach involves providing the network with both inputs and outputs, enabling the network to comprehend the type of link between input and output. The network adjusts the weight coefficient values it assigns as it gains knowledge of the relationship between input and output until the difference between the estimated value and the actual output falls to a predetermined level. The error value for ANN refers to the difference between the network's estimated value and its actual output. The smaller this value, the closer the network predicted output value will be to the actual output value. The coefficients assigned by the network are changed according to certain learning rules. MLP updates the weighting coefficients according to the "Generalized Delta Rule" learning rule. Forward calculation and backward calculation are the two stages of the Generalized Delta Rule. Feed-forward backpropagation ANN calculates output against given input while feeding forward. Neurons in the input layer are connected to the hidden layer with certain weight connections to the output layer, with certain weight connections in the hidden layer. After the data in the input layer is multiplied by the weights to which they are connected and collected in the addition function, it passes through the activation function to the hidden layer, in the same way, the data received from the input layer in the hidden layer is multiplied by the weights they are connected to, and is collected in the addition function and sends it to the output layer as output data. Usually, the preferred activation function for MLP is the sigmoid function.

The equation (1) of the addition function is as follows.

$$Net = \sum_{i=1}^{n} w_{ij} x_i + b_j \quad (1)$$

where; x_i is the input value of neuron ($i = 1,2,\ldots,n$), w_{ij} is the weight coefficient, n is the overall number of inputs going to a neuron and b_j is the Bias value. b_j is a threshold value that the Net value must surpass to generate an outcome. Usually, threshold value neurons assigned as -1 or +1 are assigned as input values [28]. However, the threshold input does not have to be assigned to ANN. In MLP, which is the most widely used today, the tangent-sigmoid function is used as the activation function in this study.

While calculating the network output in the forward calculation phase, the weight coefficients are updated in the reverse calculation phase. In the backpropagation learning algorithm, there is a forward flow of information between the layers, while backward error spreads so that the total square error is minimized. In this context, backpropagation algorithms have been developed to minimize the specified performance function. Two of the most used backpropagation algorithms when training ANN are Levenberg-Marquardt and Bayesian Regularization algorithms. With these algorithms, in order to bring the predicted data of the network as close as possible to the actual data, that is, to minimize the error value, the weight coefficients are changed, and the estimated outputs are recalculated until they fall below a certain value. In this study, the existing flows of the Karasu River have been estimated by using the feed-forward back-propagation ANN model and used two backpropagation algorithms when training ANN are Levenberg-Marquardt and Bayesian Regularization algorithms. With these algorithms, in order to bring the predicted data of the network as close as possible to the actual data, that is, to minimize the error value, the weight coefficients are changed, and the estimated outputs are recalculated until they fall below a certain value. Levenberg-Marquardt and Bayesian Regularization training algorithms are employed and their performances are compared in this study.

2.2.1. Levenberg-Marquardt Algorithm

Levenberg-Marquardt algorithm, which removes the constraints of Gauss-Newton and gradient-descent algorithms and consists of the best features, is a least-squares calculation method. It is a simplified version of the classical Newton method used in training MLP. The performance function can be taken as mean squared error (Equation (2)) which given below.

$$E_d = MSE = \frac{1}{n}\sum_{i=1}^{n}(T_i - Y_i)^2) \quad (2)$$

where; T_i is the expected value, Y_i is the output and E_d is the mean squared error of the network.

The Jacobian matrix, $J(w)$, is obtained from the first derivatives of the network errors according to the weights. At the backpropagation stage of the network error, firstly, the gradient of the network, *G(w)* is computed by using the transposition of the Jacobian matrix and the network errors (Equation (3)).

$$G(w) = J^T(\mathrm{w})\, e(w) \quad (3)$$

where; e is the error vector. After calculating the gradient of the network, the vector change in the weights of the network, $\Delta w = w_{new} - w_{old}$, is determined by multiplying the inverse of the Hessian matrix (Equation (4)) with the gradient of the network.

$$H(w) = J^T(w)J(w) + \mu I \quad (4)$$

where; μ is the Marquardt parameter, I is the Unit matrix and w is the weight vector.

While the network is trained with the Levenberg-Marquardt algorithm and minimizing the performance function with respect to weight vector, the weight change of the network is calculated as in Equation (5).

$$\Delta w = -[H(w)]^{-1}G(w) \quad (5)$$

where $H(w)$ is the Hessian matrix and $G(w)$ is the gradient.

μ parameter is identified as a numerical number for the Levenberg-Marquardt algorithm. The process continues to work as Newton's algorithm if μ is getting closer to zero; if μ is enlarging, the algorithm switches to the gradient reduction method [29], [30]. Newton's method is more rapid and precise when it is close to a minimum error. Therefore, the goal becomes switching to Newton's method at the earliest time. μ decreases when reduction in performance function occurs and only increases when there is an increment in the performance function depending on a decay rate, thus, each time the algorithm iterates, the performance function always declines [31].

2.2.2. Bayesian Regularization Algorithm

Unlike the traditional neural network backpropagation, which adjusts the optimum weight coefficients by minimizing the error function, the Bayesian regularization algorithm uses the probability distribution of the network's weights [32]. In other words, the estimates made by the network are based on probability distribution. In training with Bayesian Regularization algorithm, the weight and bias values are refreshed according to Levenberg-Marquardt optimization. Bayesian Regularization approach, which automatically arranges the appropriate performance function to achieve successful generalization, was developed by MacKay (1992) [33]. Large-value weights can cause output to vary excessively, and Regularization is the traditional method of addressing the negative impact of large-value weights. Bayesian Regularization approach includes probability distribution of network weights. Consequently, the estimates made for the network are also a probability distribution. Bayesian Regularization involves modifying the performance function commonly used, such as the sum of mean square errors (MSE). Bayesian Regularization algorithm aims to enhance the model's capacity for generalization of the model. In the training phase, the E_w term, which is the sum of the squares of the performance function net weights is expanded to improve the generalization ability of the network (Equation (6)) [32];

$$F = \beta E_d + \alpha E_w \quad (6)$$

where; E_w is the sum of squares of the network weights and F is the regularized objective function. The α and β parameters need to be estimated and adjusted according to the Bayesian Regularization algorithm. If $\alpha << \beta$, the Bayesian Regularization training algorithm shrinks errors further. If $\alpha >> \beta$ will emphasize the reduction of training weight size, thus producing a smoother network response [34] and decreasing chance of overfitting with better generalization. Adjusting the proper values for α and β parameters is the key challenge in regularization implementation [34]. Overfitting may be inevitable if α is too big and the network does not properly fit the training data if it is too small.

According to the Bayesian Regularization rule, the posterior distribution of the weights of ANN can be updated using Equation (7):

$$P(w|D,\alpha,\beta,M) = \frac{P(D|w,\beta,M)\times P(w|\alpha,M)}{P(D|\alpha,\beta,M)} \quad (7)$$

where; M is the specific ANN architecture used and D is the training set consisting of input and target data. In the implementation of the Bayesian Regularization algorithm, optimum weights should maximize the posterior probability, $P(w|D,\alpha,P,M)$, because maximizing the posterior probability of the weights corresponds to minimizing the regularized objective function (Equation 7) [34], [35].

Foresee and Hagan (1997) [34] put forward the procedure to achieve optimum values of α and β parameters:

Firstly; α, β and the weights are initialized. Initially, α is selected as 0.

Then, one step of the Levenberg-Marquardt algorithm is taken to minimize the regularized objective function, F.

The effective number of parameters, γ, which measures how many network parameters, such as weights and biases, are utilized by the neural network to minimize the error function, is calculated.

New predictions for α and β parameters are executed.

Lastly, iterations from step 2 to step 4 is performed until convergence.

3. Performance Measures

Performance evaluation was carried out with the root mean squared error (RMSE), mean absolute error (MAE), Percent Bias (PBIAS), Nash-Sutcliffe efficiency (NSE), and the coefficient of determination (R^2).

The standard deviation of the prediction errors can be identified as Root Mean Square Error (RMSE). In forecasting and regression analysis studies, root mean square error, RMSE, is frequently used to validate experimental results.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(T_i - Y_i)^2}{n}} \quad (8)$$

The average absolute variance between the observed and the estimated values is referred as the mean absolute error (MAE). It is not considered to examine under and overestimation and changes linearly. Like RMSE, it is a preferable metric in forecasting studies.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|T_i - Y_i| \quad (9)$$

Both RMSE and MAE show that the closer the error values are to 0, the more predicted values approach to the expected values.

Underestimation or overestimation of the forecast is quantified by the bias ratio. According to Gupta et al. (1999) [36] and Moriasi et al. (2007) [37], positive PBIAS indicates model underestimation bias and negative PBIAS shows model overestimation bias.

$$PBIAS = \frac{\sum_{i=1}^{n}(T_i - Y_i)*100}{\sum_{i=1}^{n}(T_i)} \quad (10)$$

Prediction ability of hydrological models is frequently evaluated using the Nash-Sutcliffe efficiency statistic [38]. According to theory, the NSE statistic ranges from -∞ to 1, with 1 denoting the ideal model.

$$NSE = 1 - \left[\frac{\sum_{i=1}^{n}(T_i - Y_i)^2}{\sum_{i=1}^{n}(T_i - \bar{T})^2}\right] \quad (11)$$

A measurement of the linear correlation between two quantities is the coefficient of determination, R^2. The coefficient of determination R^2 is defined as the square of the correlation coefficient. The determination coefficient, the R^2 values demonstrate how well the forecasted and observed values correspond and it is a value ranging from 0-1, the closer to 1, the more predicted values converge to the real results.

$$R^2 = \left(\frac{n\sum_{i=1}^{n} T_i Y_i - (\sum_{i=1}^{n} T_i)(\sum_{i=1}^{n} Y_i)}{\sqrt{\left[n(\sum_{i=1}^{n} {T_i}^2) - (\sum_{i=1}^{n} T_i)^2\right]\left[n(\sum_{i=1}^{n} {Y_i}^2) - (\sum_{i=1}^{n} Y_i)^2\right]}}\right)^2 \quad (12)$$

where $\bar{T}$ denotes the mean of experimental findings.

By considering Moriasi et al. (2007) [37], the performance is determined in this study as follows:

0.75<NSE≤1: “very good”; 0.65<NSE≤0.75: “good”; 0.50<NSE≤0.65: “satisfactory”,

PBIAS<±10%: “very good”; ±10% ≤PBIAS <±15%: “good”; ±15%≤PBIAS<±25%: “satisfactory”.

4. Results and Discussion

This study presents modelling of the rainfall-runoff process of D21A001 Kırkgöze gauging station on the Karasu River by using the artificial intelligence method, MLP. ANNs were created by updating the code produced by Neural Network Toolbox Fitting Tool in the MATLAB environment. Training, validation, and testing sets of data were created. The first 70% of the data series were chosen to serve as the training set, the next 5% as the validation set, and the final 25% as the testing set for the Levenberg-Marquardt algorithm. For the Bayesian Regularization algorithm, the first 75% of the data series were chosen as the training set and the next 25% of the data series were picked as the testing set. The tangent-sigmoid function was utilized as an activation function. Input runoff and rainfall scenarios were evaluated. Following the creation of several network designs, for training, it was decided on the scenario inputs and the quantity of neurons in the hidden layer. In order to make an evaluation for the ANN which gives the best result, some performance functions were utilized, i.e., the minimum root mean square errors, RMSE and the maximum determination coefficients, R^2 for the testing period. Then, accordingly, the scenario inputs and the number of the neurons in the hidden layer used in this study were investigated.

The scenarios were created using the past and current rainfall, temperature which is presented in Table 2. The inputs include the previously observed daily discharge(Q) and rainfall (P) and temperature (T), (Q(t-1),Q(t-2),Q(t-3), P(t-1),P(t-2),P(t-3), T(t-1),T(t-2), and T(t-3)), and the output is assigned as the current runoff (discharge) (Q(t)) (t is the current time).

First, all the scenarios were trained to use as odd numbers from 3 to 21 hidden neurons, their performance was examined, and the best 6 were selected for further calculations. As a result, 6 scenarios including a variety of inputs of Q, P, and T data were presented in Table 2 and both of the training algorithms were applied in modelling of rainfall-runoff to identify the ideal scenarios.

Table 2: Various input model scenarios.

Scenarios	Inputs	Output
1	P(t-1),T(t-1),Q(t-1)	Q(t)
2	P(t-2),T(t-2),Q(t-2)	Q(t)
3	P(t-3),T(t-3),Q(t-3)	Q(t)
4	P(t-1),Q(t-1)	Q(t)
5	Q(t-1)	Q(t)
6	Q(t-1),Q(t-2)	Q(t)

Training took place in accordance with the training parameters. The μ was increased by increase factor for μ until the change in performance reached a reduced performance value. Then, the change was performed to the network and μ was decreased by the decrease factor for μ. When the maximum number of epochs was reached, the performance gradient fell below the minimum performance gradient, the performance was minimized to the target, or μ exceeded the maximum value for μ, training was terminated. Besides, while training occurred with Levenberg-Marquardt algorithm, training stopped when validation performance has increased more than maximum validation failures since the last time it decreased while using validation. However, validation stops were not utilized by arranging maximum validation failures as infinite for Bayesian Regularization algorithm so that training was able to proceed until an optimal combination of errors and weights were obtained.

Maximum number of epochs to train was taken maximum as 1000, performance goal for all the data as 0. For Levenberg-Marquardt training algorithm, the number of maximum validation failures was taken as 6. Maximum time to train in seconds was taken as infinite, minimum performance gradient as 10^{-7}. For Levenberg-Marquardt algorithm, initial μ was taken as 0.001 and for Bayesian Regularization Algorithm; Marquardt adjustment parameter was taken as 0.005. The decrease factor for μ was as 0.1, increase factor for μ as 10, maximum value for μ as 10^{10}.

For activation function, tangent-sigmoid was used and performance function was taken as mean squared error, MSE, function during the process. The learning rate was arranged as 0.01 and the momentum constant as 0.9 for gradient descent with momentum weight and bias learning function. During the process of creating ANNs, the input and target data were normalized from its original range to the range [-1, 1].

For both algorithms, the number of hidden neurons was chosen as the odd numbers from 3 to 21, respectively, and 50 independent ANNs were created for each selected hidden number of neurons. When the number of hidden neurons has been taken more than 21, it has been observed that there became a decline in the performance of ANN. That's why; the maximum number of neurons was taken as 21. The mean squared error (MSE) and determination coefficient (R^2) results of the 50 created independent ANNs were averaged, and the best-hidden neuron number was selected considering the test results of the smallest mean MSE and largest mean R^2 values of these 50 ANNs. This process was repeated for each scenario. Then, conducted 50 ANNs according to the best selected hidden neuron number were examined for each scenario and by taking into account the test results of the network with the smallest RMSE, MAE, and PBIAS with the largest NSE and R^2 values between each 50 ANNs, the best scenario was selected. For RMSE, MAE and PBIAS, lower numbers are preferable, whereas NSE and R^2 is better with levels near 1.

Overfitting is one of the issues that arise during the training of neural networks. When the network is provided with new data, the error is much larger than it is when the training set is used. The network internalizes the training samples; however, it is incapable of generalizing to unexpected situations. The regularization and early stopping techniques can be considered as two ways to enhance generalization.

Three selected sets of the present data are used in the early stopping method. The training set, which is the first set, is utilized to compute the gradient and update the weights and biases of the network. Validation set composes the second set. Throughout the training process, the validation set error is tracked. Both the training set error and the validation set error commonly decrease during the first stage of training. Nonetheless, the validation set error can begin to increase when the network starts to overfit the data.

Early stopping method was applied to prevent the network from becoming overfit to the training data set while training with Levenberg-Marquardt algorithm. The training was terminated and the weights and biases with the smallest validation error were returned after the validation error increased for a predetermined number of iterations. The number of maximum validation failures is a measure of subsequent iterations during which the validation performance does not improve. The training came to end when this predetermined number of iterations achieved 6.

The value of the performance function was plotted against the number of iterations in the performance plot. Performances throughout training, validation, and testing were depicted. The iteration at which the least validation performance achieved was shown with the best epoch. Six additional iterations were executed until training was terminated.

It can be pointed that there is no significant problem with training according to Figure 3. The validation and test curves are very similar. It is likely that some overfitting may have taken place if the test curve had increased sufficiently before the validation curve did.

This graph demonstrates how training and validation errors decrease until the epoch indicated. Since the validation error did not grow prior to epoch 5, overfitting does not seem to have been place.

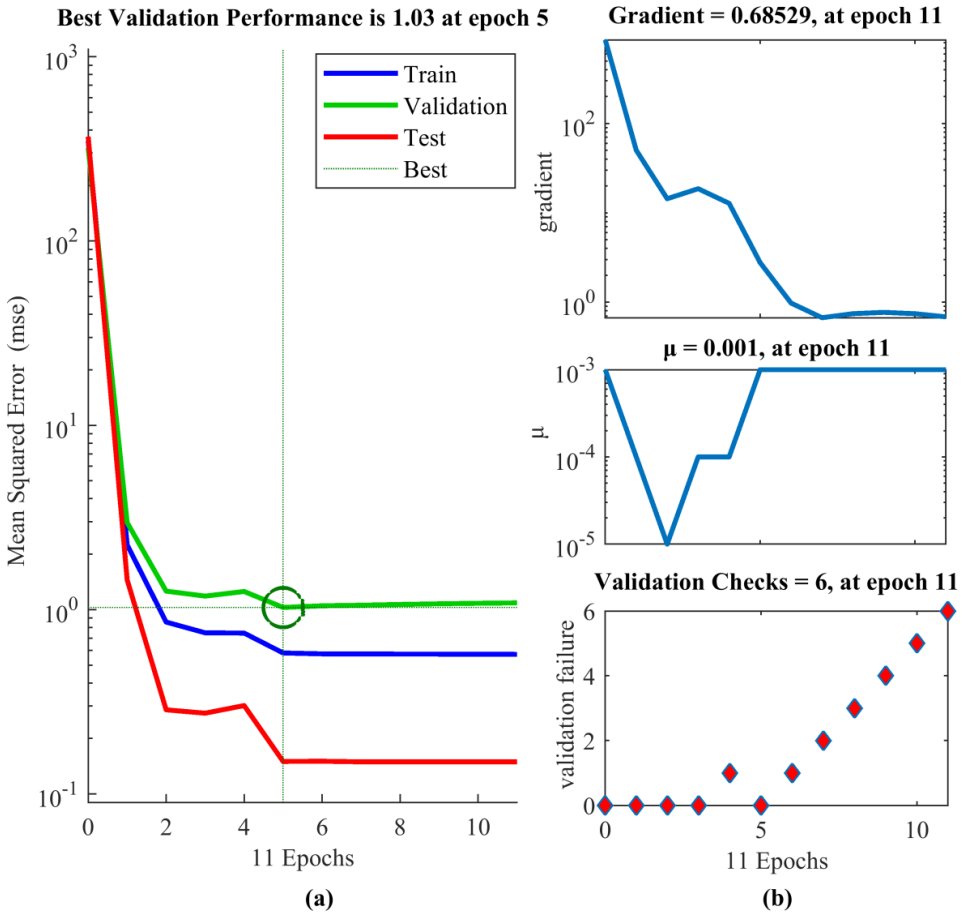

Figure 3: a) Performance curves and b) training states for Levenberg-Marquardt.

Regularization is the other method for improving generalization. It might be used instead of validation during training in order foster effective generalization. Validation is typically employed as a sort of regularization; however, training with Bayesian Regularization has its own sort of validation integrated into the algorithm. There is no validation check during training with Bayesian Regularization so that no validation set since the goal of verifying validation is to see whether the validation set error improves or worse over time. Thus, it is possible to keep training until the optimal combination of errors and weights is discovered. The errors observed while training with Bayesian Regularization not only originating from the performance of the model, but also from the weights because greater weights result in higher error. That's why; determining a number of maximum validation failures can prevent the network to experience greater weights so that prevent searching for an ideal combination of squared errors and weights with increasing number of iterations.

It's crucial to train the network until convergence when using Bayesian regularization. The sum squared weights (SSW) and the sum squared error (SSE) should all achieve constant values after multiple iterations once the network has converged [39]. Besides, the algorithm should be allowed to run until the effective number of parameters, γ, converges without considering how many parameters there are in the network [39]. If training is terminated when μ reaches maximum, the algorithm will be in fact converged.

It can be seen that the effective number of parameters, γ, and sum squared weights achieved constant values after multiple iterations and performance curves were converged (Figure 4). The best training performance was obtained at epoch 374.

Eventually, while training stopped due to reaching maximum validation failures at epoch 11 in Levenberg-Marquard and since maximum μ was reached at epoch 491 in Bayesian Regularization. Table 3 and Table 4 give performance evaluation of Levenberg-Marquardt and Bayesian Regularization algorithms through the R^2, NSE, RMSE, MAE and PBIAS indexes.

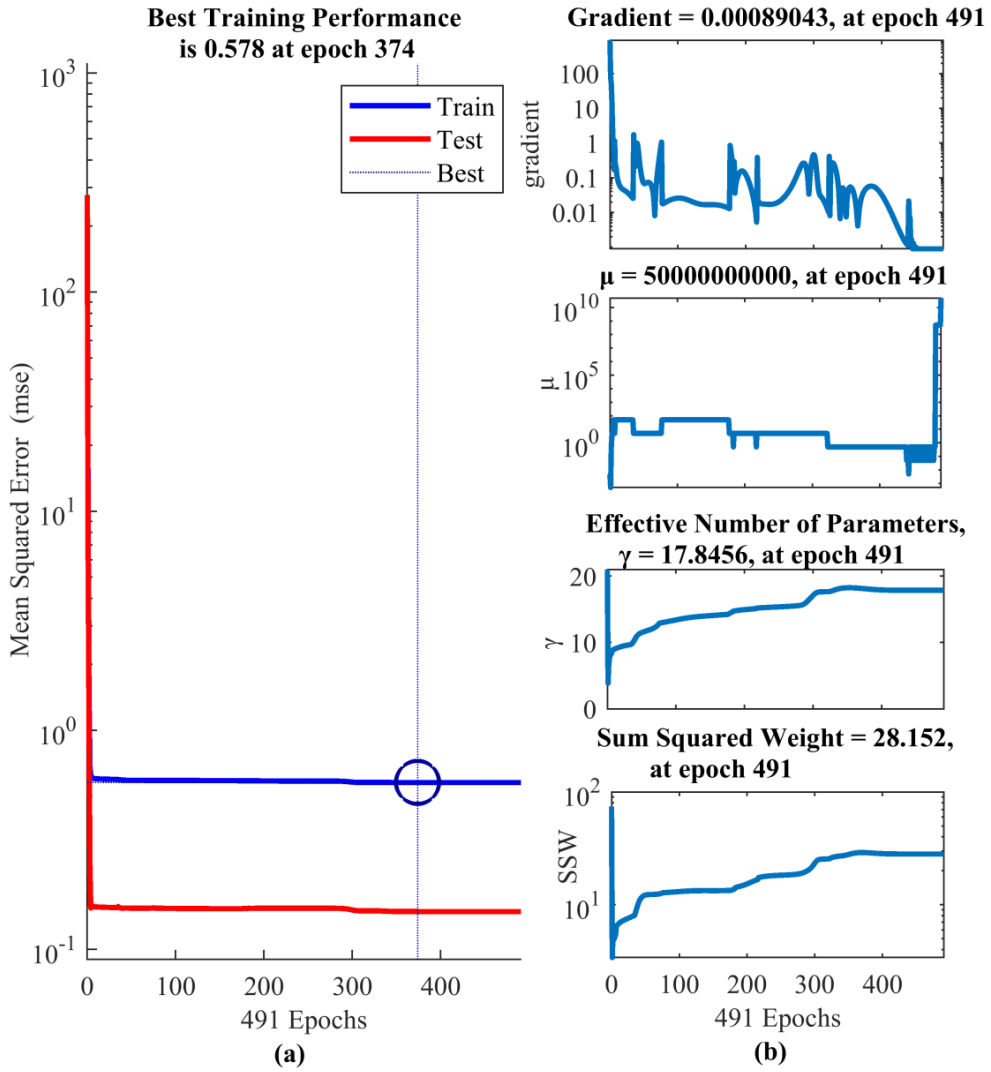

Figure 4: a) Performance curves and b) training states for Bayesian Regularization.

Table 3: Testing statistics of the LM according to the best hidden neuron number.

Levenberg-Marquardt (LM)											
Scenario	**Number of Hidden Neurons**	**Training**					**Testing**				
		RMSE (m^3/s)	**MAE (m^3/s)**	**PBIAS (%)**	**NSE**	**R^2**	**RMSE (m^3/s)**	**MAE (m^3/s)**	**PBIAS (%)**	**NSE**	**R^2**
1	3	0.738	0.323	0.34	0.95	0.95	0.391	0.202	2.42	0.97	0.97
2	9	0.982	0.473	1.16	0.91	0.91	0.572	0.306	4.44	0.93	0.93
3	13	1.203	0.585	-5.81	0.86	0.86	0.674	0.358	-1.75	0.90	0.90
4	3	0.762	0.315	0.001	0.94	0.94	0.397	0.186	1.11	0.96	0.97
5	13	0.762	0.321	0.03	0.94	0.94	0.401	0.189	1.41	0.96	0.96
6	**3**	**0.763**	**0.303**	**-1.96**	**0.94**	**0.94**	**0.387**	**0.177**	**-1.08**	**0.97**	**0.97**

Table 4: Testing statistics of the BR according to the best hidden neuron number.

Scenario	Bayesian Regularization (BR)										
	Number of Hidden Neurons	Training					Testing				
		RMSE (m^3/s)	MAE (m^3/s)	PBIAS (%)	NSE	R^2	RMSE (m^3/s)	MAE (m^3/s)	PBIAS (%)	NSE	R^2
1	15	0.692	0.306	0.00003	0.96	0.96	0.386	0.200	2.15	0.97	0.97
2	17	0.975	0.463	0.01	0.92	0.91	0.571	0.297	3.19	0.93	0.93
3	9	1.206	0.586	0.0003	0.87	0.87	0.677	0.360	3.68	0.90	0.90
4	3	0.773	0.323	0.0002	0.95	0.95	0.399	0.188	1.18	0.96	0.96
5	3	0.792	0.324	0.01	0.94	0.94	0.402	0.186	1.22	0.96	0.96
6	**5**	**0.760**	**0.313**	**-0.02**	**0.95**	**0.95**	**0.386**	**0.184**	**1.48**	**0.97**	**0.97**

Networks created with rainfall and temperature data alone did not work effectively, thus they were not added to the tables. 50 independent networks were created for each hidden number of neurons and the best numbers of hidden neurons were chosen according to their average performance values.

This is because each network is different from others, although all are created with the same hidden number of neurons. Because during the creation of each network, there are changes in the weight and threshold values given to the network, therefore, the same network is not created for the same number of hidden neurons, and the performance of each network changes. However, it has been observed that the error values and performance values of the ANNs created according to the number of hidden neurons are similar, but not exactly the same, so 50 networks are created, and their performance is averaged, and the results of the overall performance are taken into account.

Additionally, each ANN model's structures are provided in Table 3 and Table 4 with their hidden layer counts. It is clear from Table 3 that the Q(t-1), Q(t-2) scenario with 3 hidden neuron numbers has the lowest RMSE, MAE, PBIAS and the largest NSE and R^2 values taking into account the test results for Levenberg-Marquardt algorithm.

Accordingly, it is seen that the MLP trained by the Levenberg-Marquart algorithm has a successful performance in the current flow estimation considering the R^2 results. It can be said that the best scenario for the application for the Bayesian Regularization algorithm is the Q(t-1), Q(t-2) scenario with 5 hidden neuron numbers has the lowest RMSE, MAE and largest NSE and R^2 values. The results show that the Bayesian Regularization algorithm can produce formulas that are both well fitted to the data and have very low mean errors. However, when scenarios were conducted by taking only precipitation and temperature data as input, it has been obtained a low performance of for both the Bayesian Regularization and Levenberg-Marquardt algorithm which are not shown in this study.

Figures 5 and 6 display the ideal ANN's observed and predicted runoff values (created by the Levenberg-Marquart and the Bayesian Regularization algorithms) during both training, validation and test periods by using scatter diagrams and continuous graphs, respectively.

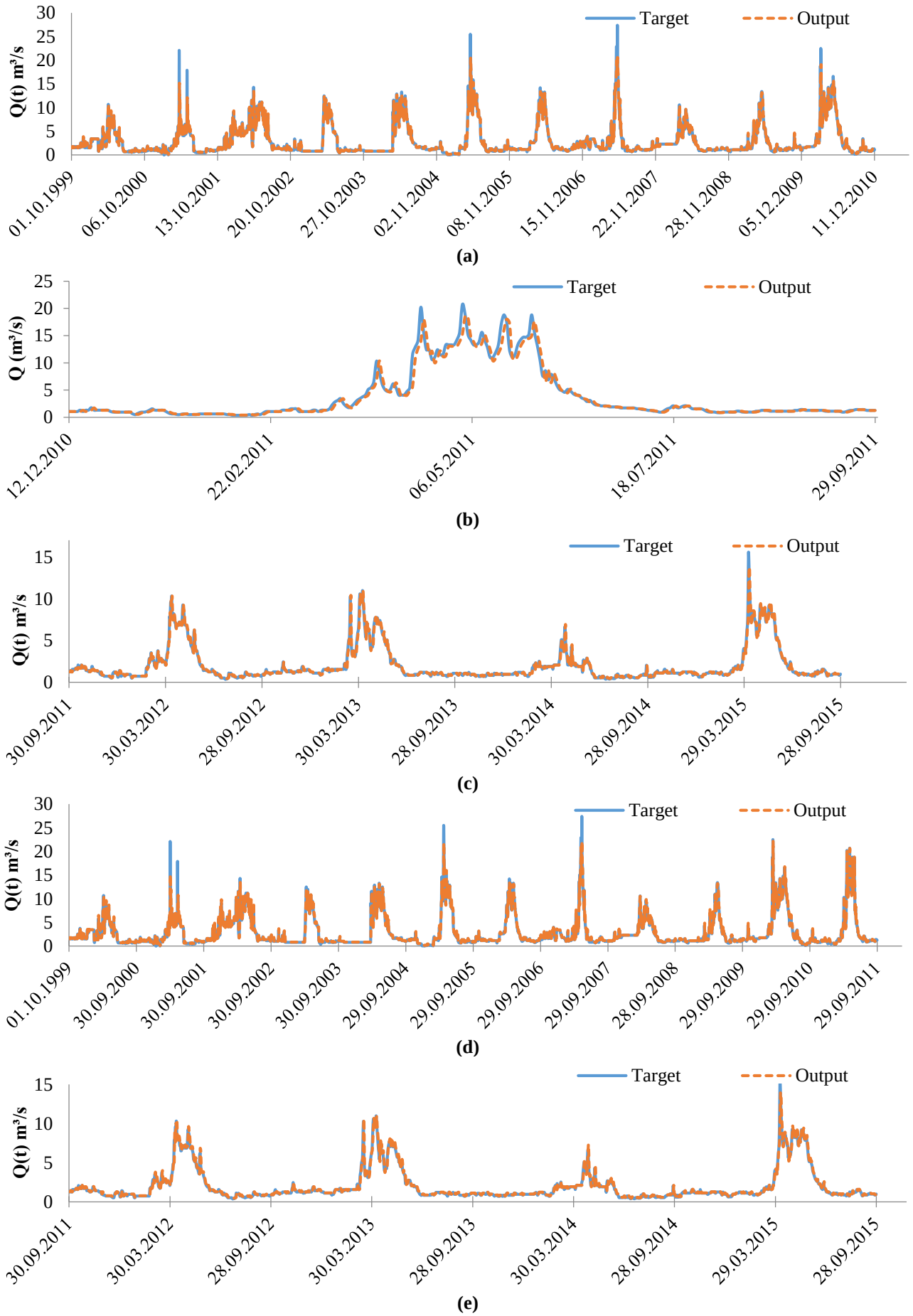

Figure 5: Observed and predicted runoff (discharge) values for scenario 6 using both LM for a) training, b) validation and c) testing cases and BR for d) training and e) testing cases.

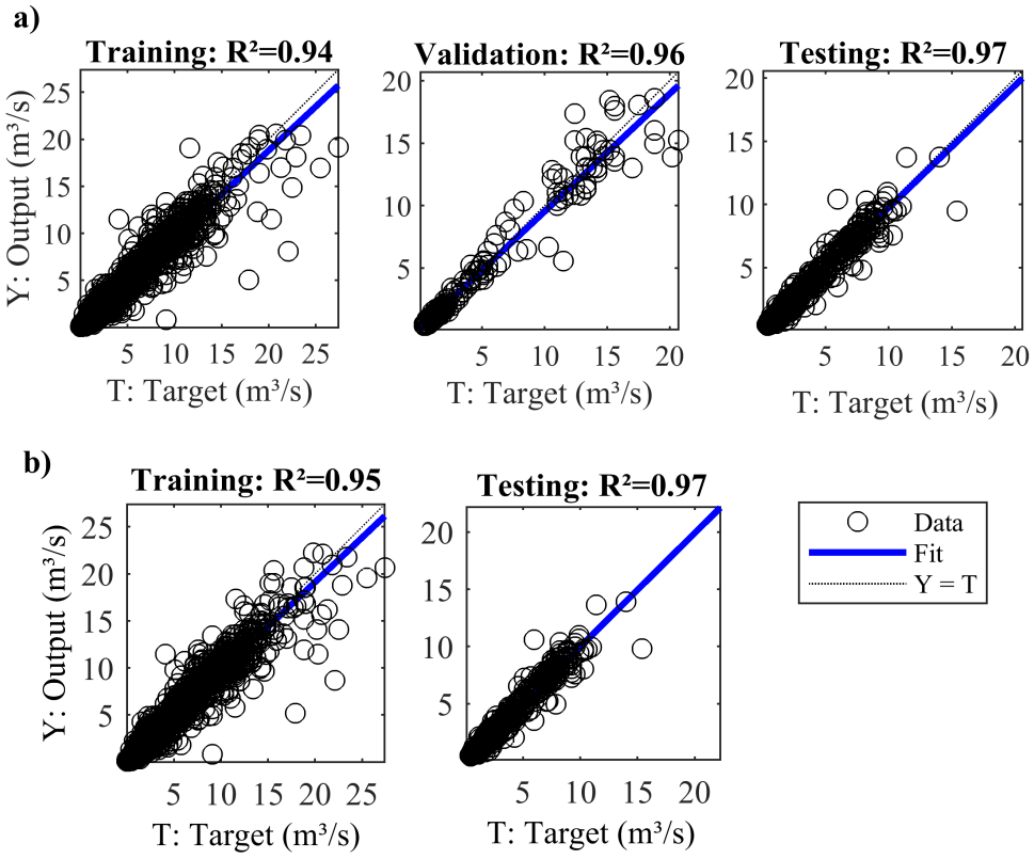

Figure 6: The scatter plots of scenario 6 for both a) LM and b) BR algorithms for training, validation and testing cases.

To more clearly investigate the performance of the training algorithms used in this study, a series of graphs in Figure 5 shows the target and output values for these clusters. The situation is shown by the diagonal line connecting the expected and observed values where the predicted values differ slightly from the observed ones. In fact, overlapping is not impossible to achieve, practically. However, the data points build up around this line, even for larger data. The performances of the same scenarios of the two algorithms are generally close to each other. In both algorithms, it has been observed that, when the flow inputs are added to scenarios with only rainfall and temperature inputs, the current day's current forecast performance is significantly increased.

According to evaluations by NSE and PBIAS, performance ratings can be considered as very good. The comparison of the findings in Table 5 demonstrates that ANN model created by the Bayesian Regularization algorithm and Levenberg-Marquart algorithm shows close performance in term of RMSE, MAE and PBIAS as well as NSE and R^2 indexes.

Table 5: The performance comparison of ANN (created by the LM and BR algorithms).

Method	Training					Testing				
	RMSE (m^3/s)	MAE (m^3/s)	PBIAS (%)	NSE	R^2	RMSE (m^3/s)	MAE (m^3/s)	PBIAS (%)	NSE	R^2
ANN-LM	0.763	0.303	-1.96	0.94	0.94	0.387	0.177	-1.08	0.97	0.97
ANN-BR	0.760	0.313	-0.02	0.95	0.95	0.386	0.184	1.48	0.97	0.97

These best ANN models for the Levenberg-Marquart and the Bayesian Regularization algorithms have R^2 and NSE values of 0.94 and 0.95 for training, 0.97 and 0.97, for testing, respectively. Besides, when RMSE values are compared, the Levenberg-Marquart and the Bayesian Regularization algorithms have 0.763 m^3/s and 0.760 m^3/s for training and 0.387 m^3/s and 0.386 m^3/s, respectively. It is seen that these values are too close. For MAE and PBIAS, there are insignificant differences.

5. Conclusion

Based on the study of rainfall-runoff modelling using ANN, the following conclusions can be drawn:

First, all scenarios were trained to use as odd numbers from 3 to 21 hidden neurons and their performance was compared. 6 scenarios with the best performance were selected, and the calculations continued over these 6 scenarios. For both ANN algorithms, the number of hidden neurons was chosen as odd numbers from 3 to 21, respectively, and 50 different ANNs were created for each selected hidden number of neurons. The reason was that there is a change in the weight and threshold values given to the network while the studies are being carried out and each network is being formed, so the same network is not created for the same number of hidden neurons, and it was discovered that the performance of each network changes. However, it is taken into account that the error values and performance values of ANNs created according to the number of hidden neurons are similar, but not exactly the same, therefore the performance of 50 networks is averaged and the overall results are obtained.

A major factor affecting the precision of model prediction is the selection of training and testing data. The model will not be able to make accurate future forecasts if the testing data don't accurately reflect basin and climate characteristics. Looking at the ANN scenarios trained with two different algorithms, it was observed that temperature and rainfall data alone were insufficient in estimating the current runoff data. It was reached that the ANN gives much improved performance by adding the past time runoff data to the inputs. This result was observed during the trial-and-error stage of our research, and scenarios related only to the past time runoff data were also added. In addition, it can be said that adding both past time rainfall and past time temperature data as input to the past time runoff, does not considerably improve the current day runoff (discharge) data estimation.

By comparing the results, the application of the used models to the rainfall-runoff process was indicated as successful. Bayesian Regularization and Levenberg-Marquardt algorithm models have close performance.

Evaluate all scenarios individually; it was reached that the higher the number of hidden neurons increases, in general, the better the Bayesian Regularization algorithm trains the networks. However, it was observed that the performance of the network decreased while testing the case. The Levenberg-Marquardt algorithm has advantages such as working with less iteration, and in less time. However, although the Bayesian Regularization algorithm worked for longer with more iterations, the best scenario yielded better results while training and testing the network. However, comparing the results of the scenarios of the two different training algorithms, it can be concluded that the values are generally close.

When looking at the graphs of the predicted and observed runoff data among the 2 methods used in this study, it can be said that the performance of the estimation of the suddenly increasing data is less successful than the performance of the estimation of the normal progressing data, but that the estimates can be improved with the application of these methods. In addition, comparing scenarios 1, 2, and 3 for all 2 methods, it is seen that the nearer in time the rainfall, temperature and runoff data is to the time of the current to be predicted, the better the predictions for each method.

As a result of the study, for the best scenario of the ANN, the past runoff data is decisive, and there is no rainfall and temperature data considered.

In this study, the ANN models can work well to forecast small number of time step ahead values and it executes a good performance as indicated by performance statistics. However, for long forecasting periods, some new scenarios can be investigated since input includes the data just one or two steps back. For large number of time step ahead forecasting, the models used in this study will not be efficient since predicted values will be needed to utilize and forecasted runoff values will include uncertainties.

It is preferable to use ANN models for applications since they can recreate hydrological models through experience-based learning. Compared to traditional regression analysis, these models are better able to forecast flood discharges. Depending on the results, we recommend that ANN models be used in the modelling of rainfall-runoff data, and flood forecasting.

Article Information

Financial Disclosure: The author (s) has no received any financial support for the research, authorship or publication of this study.

Authors' Contrtibution: Concept: Bor; Design: Bor, Okan; Supervision: Bor; Resources: Bor; Data Collection: Bor; Analysis: Okan; Literature Search:

Okan; Writing Manuscript: Bor, Okan; Critical Review: Bor.

Conflict of Interest/Common Interest: Not applicable.

Ethics Committee Approval: Not applicable**.**

Declaration of the Author(s): The author(s) declare that there is no conflict of interest regarding the publishing of the paper by the Journal of Innovative Science and Engineering, that the paper has been not published elsewhere, and not include any form of plagiarism. All the authors listed above have approved the manuscript and have agreed with the submission of the manuscript to the Journal of Innovative Science and Engineering.

Acknowledgements: The authors would like to thank the DSI (General Directorate of State Hydraulic Works), Department of Survey, Planning, and Allocations for the providing of the data.

Data availability statement: The data that support the findings of this study are available from [DSI (General Directorate of State Hydraulic Works), Department of Survey, Planning, and Allocations, Environment Branch Directorate] but restrictions apply to the availability of these data, which were used under license for the current study, and so are not publicly available. Data are however available from the authors upon reasonable request and with permission of [DSI (General Directorate of State Hydraulic Works)].

References

[1] Tan Kesgin, R. I., Demir, I., Kesgin, E., Abdelkader, M., & Agaccioglu, H. (2023). A data-driven approach to predict hydrometeorological variability and fluctuations in lake water levels. *Journal of Water and Land Development*, (58), 158-170. https://doi.org/10.24425/jwld.2023.146608.

[2] Demirel, M. C., Özen, A., Orta, S., Toker, E., Demir, H. K., Ekmekcioğlu, Ö., Tayşi, H., Eruçar, S., Sağ, A. B., Sarı, Ö., Tuncer, E., Hancı, H., Özcan, T. İ., Erdem, H., Koşucu, M. M., Başakın, E. E., Ahmed, K., Anwar, A., Avcuoğlu, M. B., Vanlı, Ö., Stisen, S., Booij, M. J. (2019). Additional Value of Using Satellite-Based Soil Moisture and Two Sources of Groundwater Data for Hydrological Model Calibration. *Water*, 11(10), 2083. https://doi.org/10.3390/w11102083.

[3] Shamseldin, A. Y. (1997). Application of a neural network technique to rainfall-runoff modelling. Journal of Hydrology, 199(3): 272–294.

[4] Tokar, S. A., and Johnson, P. A. (1999). Rainfall-Runoff Modeling Using Artificial Neural Networks. Journal of Hydrologic Engineering, 4(3): 232–239.

[5] Chang, F.-J., and Chen, Y.-C. (2001). A counterpropagation fuzzy-neural network modeling approach to real time streamflow prediction. Journal of Hydrology, 245: 153–164.

[6] Öztopal, A., Kahya, C., and Asilhan, S. (2001). Yapay Sinir Ağları ile Akış Tahmini. 1. Türkiye Su Kongresi, İstanbul, Türkiye, 8 - 10 Ocak 2001, cilt.1. pp. 311–318.

[7] Jayawardena, A. W., and Fernando, T. M. K. G. (2001). River flow prediction: An artificial neural network approach. Regional Management of Water Resources, Maastricht, The Netherlands. pp. 239–246.

[8] Sivakumar, B., Jayawardena, A., and Fernando, T. M. K. G. (2002). River Flow Forecasting: Use of Phase-Space Reconstruction and Artificial Neural Networks Approaches. Journal of Hydrology, 265: 225–245.

[9] Dorado, J., RabuñAL, J. R., Pazos, A., Rivero, D., Santos, A., and Puertas, J. (2003). Prediction and modeling of the rainfall-runoff transformation of a typical urban basin using ann and gp. Applied Artificial Intelligence, Taylor & Francis, 17(4): 329–343.

[10] Kişi, Ö. (2005). Daily River Flow Forecasting Using Artificial Neural Networks and Auto-Regressive Models. Turkish Journal of Engineering and Environmental Sciences, 29: 9–20.

[11] Demirpençe, H. (2005). Köprüçay Akımlarının Yapay Sinir Ağları ile Tahmini. Antalya Yöresinin İnşaat Mühendisleri Sorunları Kongresi.

[12] Yurdusev, M. A., Acı, M., Turan, M. E., and İçağa, Y. (2008). Akarçay Nehri Aylık Akımlarının Yapay Sinir Ağları ile Tahmini. Celal Bayar Üniversitesi Fen Bilimleri Dergisi, 4(1): 73–88.

[13] Okkan, U., and Mollamahmutoğlu, A. (2010). Çoruh Nehri Günlük Akımlarının Yapay Sinir Ağları ile Tahmin Edilmesi. Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 14(3): 251–261.

[14] Okkan, U., and Dalkilic, H. Y. (2010). Demirköprü Barajı Aylık Buharlaşma Yüksekliklerinin Yapay Sinir Ağları ile Tahmin Edilmesi. DSİ Teknik Bülten, 108: 30–36.

[15] Chen, S. M., Wang, Y. M., and Tsou, I. (2013). Using artificial neural network approach for modelling rainfall-runoff due to typhoon. Journal of Earth System Science, 122(2): 399–405.

[16] Kızılaslan, M. A., Sağın, F., Doğan, E., and Sönmez, O. (2014). Aşağı Sakarya Nehri akımlarının yapay sinir ağları ile tahmin edilmesi. SAÜ Fen Bilimleri Dergisi, 18(2): 99–103.

[17] Singh, G., Panda, R. K., and Lamers, M. (2015). Modeling of daily runoff from a small agricultural watershed using artificial neural network with resampling techniques. Journal of Hydroinformatics, 17(1): 56–74.

[18] Khan, M. Y. A., Hasan, F., Panwar, S., and Chakrapani, G. J. (2016). Neural network model for discharge and water-level prediction for Ramganga River catchment of Ganga Basin, India. Hydrological Sciences Journal, Taylor & Francis, 61(11): 2084–2095.

[19] Altunkaynak, A., and Başakin, E. E. (2018). Zaman Serileri Kullanılarak Nehir Akım Tahmini ve Farklı Yöntemlerle Karşılaştırılması. Erzincan University Journal of Science and Technology, 11(1): 92–101.

[20] Nacar, S., Hınıs, M. A., and Kankal, M. (2018). Forecasting Daily Streamflow Discharges Using Various Neural Network Models and Training Algorithms. KSCE Journal of Civil Engineering, 22(9): 3676–3685.

[21] Bor, A., and Okan, M. (2019). FIRAT HAVZASI karasu günlük akimlarinin yapay sinir ağlari ile modellenmesi. 10. Ulusal Hidroloji Kongresi, Muğla, Türkiye, Cilt 2. pp. 857-869.

[22] Fırat, M., and Dikbaş, F. (2006). Göllerde üç boyutlu hidrodinamik modellemede pom ve yapay sinir ağlari yöntemlerinin kullanilmasi : gökpinar baraj gölü örneği. Pamukkale University Engineering College Journal of Engineering Sciences, 12(1): 43–50.

[23] Coulibaly, P., Anctil, F., and Bobée, B. (1999). Prévision hydrologique par réseaux de neurones artificiels : état de l'art. Canadian Journal of Civil Engineering, 26: 293–304.

[24] Minns, A. W., and Hall, M. J. (1996). Artificial neural networks as rainfall-runoff models. Hydrological Sciences Journal, 41: 399–417.

[25] Gümüş, V., Başak, A., and Yenigün, K. (2018). Yapay Sinir Ağları ile Şanlıurfa İstasyonunun Kuraklığının Tahmini. Gazi Üniversitesi Fen Bilimleri Dergisi, 6(3): 621–633.

[26] Haykin, S. (1998). Neural Networks : A Comprehensive Foundation. Prentice-Hall. Upper Saddle River, NJ.

[27] Öztemel, E. (2006). Yapay Sinir Ağları. Papatya Publishing, Istanbul, Turkey.

[28] Şen, Z. (2004). Yapay Sinir Ağları İlkeleri. Turkish Water Foundation, Istanbul,Turkey.

[29] Chen, T. C., Han, D. J., Au, F. T. K., and Tham, L. G. (2003). Acceleration of Levenberg-Marquardt Training of Neural Networks with Variable Decay Rate. Proceedings of the International Joint Conference on Neural Networks, IEEE. pp. 1873–1878.

[30] Hagan, M. T., and Menhaj, M. B. (1994). Training Feedforward Networks with the Marquardt Algorithm. IEEE Transactions on Neural Networks, 5(6): 989–993.

[31] trainlm Levenberg-Marquardt backpropagation. https://www.mathworks.com/help/deeplearning /ref/trainlm.html [Accessed: 07 september 2023].

[32] Xu, M., Zeng, G., Xu, X., Huang, G., Jiang, R., and Sun, W. (2006). Application of Bayesian regularized BP neural network model for trend analysis, acidity and chemical composition of precipitation in North Carolina. Water, Air, and Soil Pollution, 172(1–4): 167–184.

[33] MacKay, D. J. C. (1992). Bayesian Interpolation. Neural Computation, 4(3): 415–447.

[34] Foresee, F. D., and Hagan, M. T. (1997). Gauss-Newton approximation to bayesian learning. IEEE International Conference on Neural Networks - Conference Proceedings, Houston, TX, USA, 9–12 June 1997. pp. 1930–1935.

[35] Kayri, M. (2016). Predictive abilities of Bayesian Regularization and Levenberg-Marquardt Algorithms in Artificial Neural Networks: A Comparative Empirical Study on Social Data. Mathematical and Computational Applications, 21(2).

[36] Gupta, H. V., Sorooshian, S., and Yapo, P. O. (1999). Status of automatic calibration for hydrologic models: comparison with multilevel expert calibration. Journal of Hydraulic Engineering, 4(2): 135–143.

[37] Moriasi, D. N., Arnold, J. G., Liew, M. W. Van, Bingner, R. L., Harmel, R. D., and Veith, T. L. (2007). Model evaluation guidelines for systematic quantification of accuracy in watershed simulations. American Society of Agricultural and Biological Engineers, 50(3): 885–900.

[38] Nash, J. E., and Sutcliffe, J. V. (1970). River flow forecasting through conceptual models part I — A discussion of principles. Journal of Hydrology, 10(3): 282–290.

[39] Improve Shallow Neural Network Generalization and Avoid Overfitting. The MathWorks, Inc. https://www.mathworks.com/help/deeplearning/ug/improve-neural-network-generalization-and-avoid-overfitting.html [Accessed: 07 september 2023].