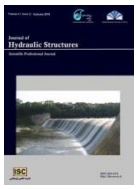
Journal of Hydraulic Structures
J. Hydraul. Struct., 2022; 8(4):73-92
DOI: 10.22055/jhs.2023.43234.1246



OPEN ACCESS

A study of five types of ANN-based approaches to predict discharge coefficient of combined weir-gate

Sanaz Afzali Ahmadabadi [1]
Arash Vatani [2]

Abstract

Since many years ago, flow measurement has become a fundamental issue in hydraulic engineering. One of the conventional methods of flow measurement is the use of combined structures. In this regard, using a combined structure, including a gate and a weir, is one of the approaches that has attracted the attention of researchers in this field. Therefore, in this research, five different methods based on artificial neural networks were used to predict the discharge coefficient. The networks architecture includes an input layer with four neurons, a hidden layer with seven neurons, and an output layer with one neuron. be mentioned that the number of neurons within the hidden layer is set to 4 only for the recurrent network. For the hidden layer, the logarithmic sigmoid activation function was used. Also, the linear activation function was used for the output layer. Finally, the results showed that the Levenberg-Marquardt (LM) algorithm performs better than the other methods. The convergence speed of this algorithm, which also uses the second derivative, is much higher than others. In this case, the coefficient of determination (R^2) for the training and the test stage was equal to 0.92616 and 0.94079, respectively. In addition to, the first type of rough model with the gradient descent training algorithm also had an acceptable performance and was placed in second place. Also, the sensitivity analysis on the dimensionless parameters affecting this issue showed that the H/d, y/d, b/B, and b/d parameters have maximum to minimum effect on the model results, respectively.

Keywords: Discharge coefficient, ANN, Flow measurement, Combined weir-gate, Hydraulic structures.

Received: 07 March 2023; Accepted: 18 March 2023

[1] School of Civil Engineering, University College of Engineering, University of Tehran, Tehran, Iran. Orcid: 0000-0001-6337-5334
[2] School of Civil Engineering, University College of Engineering, University of Tehran, Tehran, Iran. E-mail: arash.vatani@ut.ac.ir, Telephone: +989111542247, Orcid: 0000-0002-4651-0781. **(Corresponding Author)**

1. Introduction

Estimating or measuring discharge is one of the main issues in all fields related to hydraulics. Hereof, many researchers try to develop and present methods for estimating discharge. The intensity of flow or discharge is described as the volume flow per unit of time. In open channels, the flow is measured by various methods, such as using the pressure difference, velocity-area method, tracer-dilution method, hydraulic structures (flumes and weirs), etc. [1]. Weirs are one of the most straightforward hydraulic structures employed for controlling water levels and measuring the channel flow. The ease and accuracy of measuring in different flow conditions led to the design of various types of weirs [1-3].

The most important part of hydraulic research is the modeling of hydraulic structures in field conditions. In irrigation projects, like irrigation and drainage networks, gates and weirs are generally used [4]. The weir is used to pass the flow over the crest. The gate is used to get rid of the sediment in the lower part of the weir. Therefore, these two types of hydraulic structures have been widely studied [5].

The weir-gate structure combines the benefits of a weir and a gate. It controls the water head and flow rate while preventing sedimentation behind the weir, by passing sediment through the gate [6]. Ahmed [7], investigated a rectangular weir combined with a rectangular gate and determined the discharge coefficient. Negm et al. [8], analyzed the impact of geometric parameters on a combined rectangular weir and inverse triangular gate with various angles. They developed a regression equation for estimating the discharge coefficient and evaluated free flow characteristics over the combined structure. Hayawi et al. [9], studied the composite structure of a triangular weir with a rectangular gate, and the results of their studies showed that with the increase in the D/h ratio (the ratio of the height of the overflow to the water head on the overflow) the discharge coefficient decreases. Aein et al. [6], simulated four combined triangular weir-rectangular gate structures using Flow-3D, based on previous experimental research. Dimensional analyses were conducted to determine the dimensionless parameters that affect the discharge coefficient. Results indicated that Flow-3D has high simulation capabilities for this structure.

Recently, by advancing soft computing techniques in almost areas of water engineering, investigators have applied them to model and predict the hydraulic and hydrology phenomenon. Successfully in this regard, the soft computing models for modeling and predicting the discharge in rivers, scouring, and discharge coefficient of weirs have been used by researchers [4,5]. Choosing appropriate intelligent models according to different hydraulic parameters and different shapes of weirs can efficiently and accurately carry out this work [10]. For example, Bilhan et al. [11], estimated the discharge coefficient (Cd) of a triangular labyrinth side weir in a curved channel by using artificial neural networks (ANN). In their study, they determined Cd using the results of 7963 laboratory tests. The performance of the ANN model was compared with multiple nonlinear and linear regression models. Root mean square errors (RMSE), mean absolute errors (MAE), and correlation coefficient (R) statistics were used as comparing criteria for the evaluation of the models' performances. There were good agreements between the measured values and the values obtained using the ANN model. It was found that the ANN model with RMSE=0.1658 in the validation stage is superior in the estimation of discharge coefficient than the multiple nonlinear and linear regression models with RMSE of 0.2054 and 0.2926, respectively. In research by Safari and et al. [3], 113 data sets of Bos were used for the applicability of Artificial Neural Network (ANN), Gene expression programming (GEP), regression models to estimate the discharge coefficient for the rectangular broad-crested weirs.

Comparing the models showed that the ANN with the highest R2 coefficient (0.9916), lowest RMSE = 0.0012, and MAE = 0.00052 has the best discharge coefficient estimation than GEP models, regression models, and other empirical relations for the rectangular broad crested weirs. Salmasi et al. [12], and Chen et al. [2], used ANN, linear regression (LR), random forest (RF), support vector machine (SVM), k-nearest neighbor (KNN) and decision tree (DT) algorithms to predict discharge coefficients for broad crested weirs and streamlined weirs, respectively. All methods can provide a reasonable prediction for Cd [2,6].

Also, in the field of combined weir-gate, some studies have been done by researchers, in the research of Parsaie et al. [13], the discharge coefficient of weir-gate was predicated using adaptive neuro-fuzzy inference systems (ANFIS) and multilayer perceptron neural network (MLP). The sensitivity analysis of MLP and ANFIS showed that the Froude number of flow upstream of the weir and the ratio of the gate opening height to the diameter of the weir are the most influential parameters on the discharge coefficient. Parsaie et al. [5], also predicted a weir-gate discharge coefficient using ANN, SVM, and ANFIS. Assessing the performance of three models show that all of them have reasonable accuracy; however, the SVM model with a coefficient of determination (= 0.94) and root mean square of error (RMSE = 0.008) has the best performance in comparison with others. Sahib et al. [4], applied ANN to predict the discharge coefficient for a combined weir and found that their model with ten neurons was highly accurate. The sensitivity analysis was used to evaluate the performance of artificial neural networks using different numbers of valid input parameters. They showed that the ANN model has a good idea of what works best for the discharge coefficient and that the network training process is easier to understand than the traditional approach of expressing the discharge coefficient as an equation.

In this paper, a model for predicting the weir-gate discharge coefficient with the approach of using ANN capabilities is presented. For this purpose, five different learning methods, including Emotional Learning (EL), Levenberg-Marquardt (LM), Flexible ANN, Rough (Type1), and Recurrent (Elman), are used for this research. Finally, the capacities of these algorithms and methods are compared to each other, and at last, for the best approach, sensitivity analysis is performed on the input parameters of the model.

2. Material and Methods

Figure 1 shows a schematic of a combined weir-gate. As shown in the figure, d is the opening height of the gate, h is the free height of the flow on the overflow, H is the total upstream head of flow and h_t is the water head of the downstream.

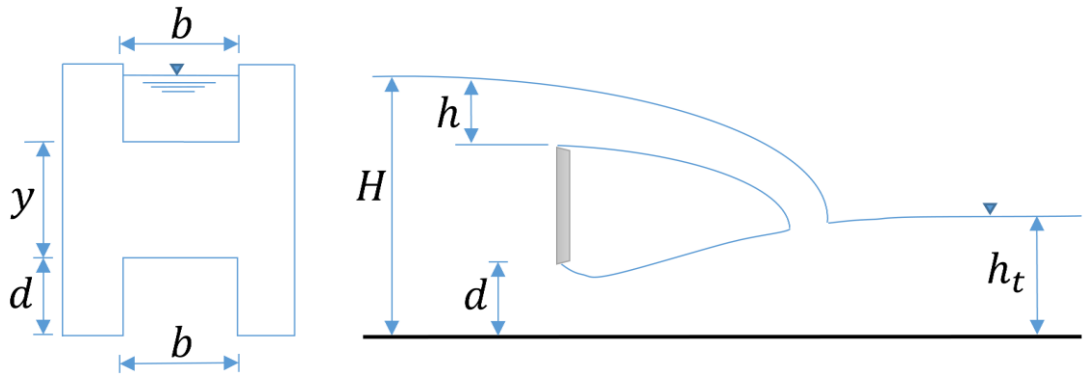


Figure 1. A scheme of H-weir-gate structure

Considering that this structure is a combination of a gate and a weir, therefore, to calculate discharge, the combination of equations governing the gate and weir is used.

$$Q_c = Q_w + Q_g \quad (1)$$

$$Q_w = \frac{2}{3} C_{dw} \sqrt{2g}\, L_e H^{1.5} \quad (2)$$

$$Q_g = C_{dg} L_d \sqrt{2g(H-d)} \quad (3)$$

Where Q_w and Q_g are the discharge of flow through the weir and gates, respectively, and Q_c is the total discharge passed from the weir-gate. Based on the dimensional analysis, the weir-gate discharge coefficient is a function of dimensionless parameters according to equation (4).

$$C_d = f\left(\frac{y}{d}, \frac{b}{d}, \frac{b}{B}, \frac{H}{d}\right) \quad (4)$$

Therefore, 4 dimensionless input parameters $\left(\frac{y}{d}, \frac{b}{d}, \frac{b}{B}, \frac{H}{d}\right)$ are required to predict the weir discharge coefficient. In this study, the data of Parsaie et al. [5], has been used, which contains 161 data. A summary of the statistical status of this dataset is presented in Table 1.

Table 1. Summaries of the statistical status of the dataset

	Min	Max	Mean	Standard Deviation
$\frac{y}{d}$	0.47	5	2.24	1.37
$\frac{b}{d}$	0.65	5	2.41	1.43
$\frac{b}{B}$	0.32	0.66	0.51	0.15
$\frac{H}{d}$	1.67	7.40	3.65	1.21
C_d	0.50	0.69	0.59	0.03

2.1. ANN

ANN is a nonlinear mathematical model that can simulate arbitrarily complex nonlinear processes, which relate to the inputs and outputs of any system. Multilayer perceptron (MLP) networks are common types of ANN widely used in research. To use the MLP model, the definition of appropriate functions, weights, and bias should be considered [5,13]. Figure 2 shows a simple configuration of a feedforward MLP model used in this study.

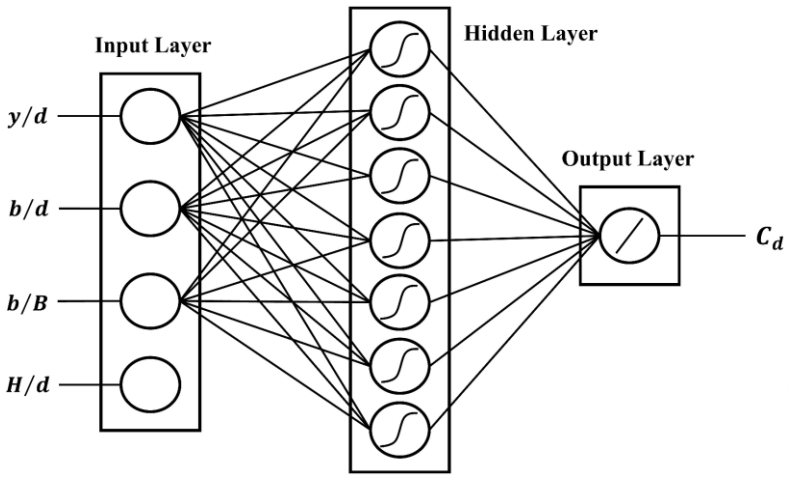


Figure 2. Multilayer perceptron neural network architecture

Weights and biases values will be adjusted dynamically during the training stage by comparing predicted output with target. Such networks are often trained using a backpropagation algorithm [5,6]. In this paper, five different methods were used to train the neural network, and their titles are listed in Table 2.

75% of the data is used for the training and 25% for the test stage. The details of the networks used, including the number of inputs, the number of hidden layers, the number of neurons of the hidden layers, and the activation functions of the first layer and the second layer, are presented in Table 2.

Table 2. Summarizes the features of the used neural network models

Model	Method	N.I	N.H.L	N.N.H.L	A.F.1	N.N.O.L	A.F.2
1	Emotional Learning	4	1	7	logsig	1	Purelin
2	Levenberg-Marquardt	4	1	7	logsig	1	Purelin
3	Flexible ANN	4	1	7	logsig	1	Purelin
4	Rough (Type 1)	4	1	7	logsig	1	Purelin
5	Recurrent (Elman)	4	1	4	logsig	1	Purelin

Notes: N.I: number of inputs, N.H.L: number of hidden layers, N.N.H.L: number of neurons in the hidden layer, A.F.1: activation function of hidden layer, N.N.O.L: number of neurons of output layer, A.F.2: activation function of output layer

2.2. Governing Equations

In this section, feedforward and backpropagation equations for several artificial neural network models used in this research are presented:

2.2.1. Two layer perceptron + Emotional Learning

Emotional learning is a training strategy for neural networks that facilitates error convergence by making it possible to use the latest information about neural parameters. In this algorithm, errors from previous times are used to learn the network, which increases the speed of convergence [14]. In this method, the error function is defined as follows:

$$E(k) = \frac{1}{2}r^2(k) = \frac{1}{2}\left(k_1 \times e(k) + k_2 \times e\cdot(k)\right)^2 \tag{5}$$

which in the above equation:

$$e(k) = Target(k) - O^2(k) \tag{6}$$
$$e\cdot(k) = e(k) - e(k-1) \tag{7}$$

that the opened form of equation (5) for a two-layer perceptron with one hidden layer can be written as follows:

$$E(k) = \frac{1}{2}(k_1 \times e(k) + k_2 \times [e(k) - e(k-1)])^2 = \frac{1}{2}\left([k_1 + k_2] \times e(k) - k_2 \times e(k-1)\right)^2 \tag{8}$$

$$E(k) = \frac{1}{2}\left([k_1 + k_2] \times (Target(k) - O^2(k)) - k_2 \times (Target(k-1) - O^2(k-1))\right)^2$$
$$= \frac{1}{2}\Big([k_1 + k_2]\left(Target(k) - f^2\left(w^2(k) \times O^1(k)\right)\right) - k_2(Target(k-1) - f^2(w^2(k-1) \times O^1(k-1)))\Big)^2 \tag{9}$$

Then updating and modifying the weights from the last layer to the first layer is done:

- Modification of weights of the last layer

$$\Delta w^2(k) = -\eta.\frac{\partial E(k)}{\partial w^2(k)} = -\eta\left(\frac{\partial E}{\partial r}\frac{\partial r}{\partial e}\frac{\partial e}{\partial O^2}\frac{\partial O^2}{\partial net^2}\frac{\partial net^2}{\partial w^2}\right)(k) = \eta.r(k).[k_1 + k_2].f'^2(k).O^1(k) \tag{10}$$

$$w^2(k+1) = w^2(k) + \eta \times r(k) \times [k_1 + k_2] \times f'^2(k) \times O^1(k) \tag{11}$$

- Modification of weights of the last layer

$$\Delta w^1(k) = -\eta.\frac{\partial E(k)}{\partial w^1(k)} = -\eta\left(\frac{\partial E}{\partial r}\frac{\partial r}{\partial e}\frac{\partial e}{\partial O^2}\frac{\partial O^2}{\partial net^2}\frac{\partial net^2}{\partial O^1}\frac{\partial O^1}{\partial net^1}\frac{\partial net^1}{\partial w^1}\right)(k) = \eta \times r(k) \times [k_1 + k_2] \times f'^2(k) \times w^2(k) \times f'^1(k) \times Input(k) \tag{12}$$

$$w^1(k+1) = w^1(k) + \eta \times r(k) \times [k_1 + k_2] \times f'^2(k) \times w^2(k) \times f'^1(k) \times Input(k) \tag{13}$$

Figure 3 shows a generalized form of the training process of a two-layer perceptron neural network based on emotional learning.

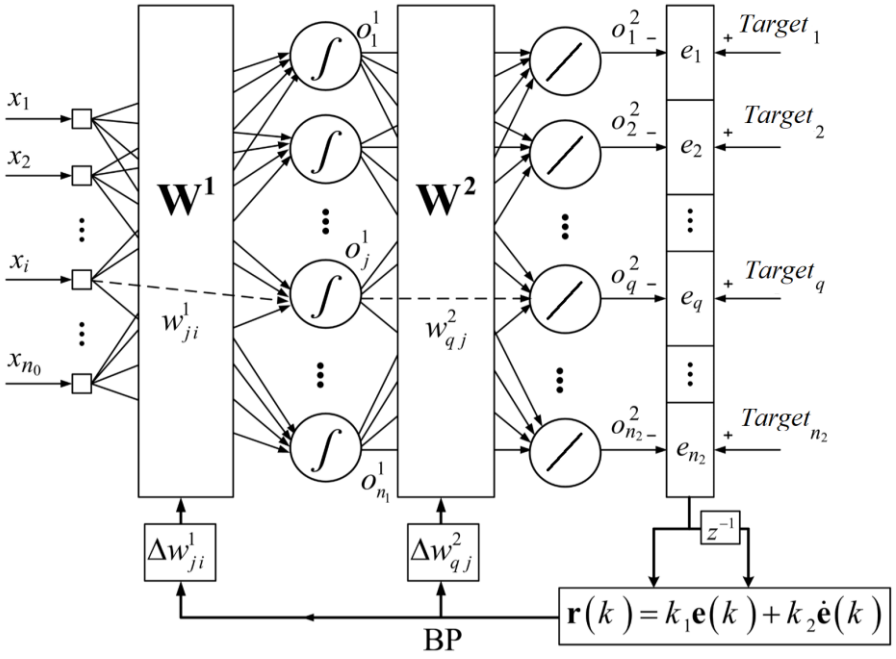


Figure 3. A two-layer perceptron neural network training process based on emotional learning.

2.2.2. Two-layer perceptron + (Levenberg-Marquardt)

Higher-order algorithms are presented to solve the problem of first-order algorithms. It has been proven that first-order algorithms are trapped in local minimum in solving many problems. In addition, due to excessive repetitions in training and adjusting the weights, they will also face problems. An alternative method for this type of problem is to use higher-order optimization methods [15].

The Levenberg-Marquardt algorithm is one of the second-order learning algorithms. In other words, in this algorithm, the second-order derivative is used to train the network, which causes fast convergence of the computational model [15]. Let the vector $w \in \mathbb{R}^{N_w}$ contain all weights or adjustable parameters in the weighted search space. Also, the target and the output of the neural network should be $Target(k) \in \mathbb{R}^M$ and $y(k) \in \mathbb{R}^M$, respectively, where M is the number of neurons in the output of the neural network. Thus, the output error of the neural network for the q-th neuron for each p-th learning pattern ($e_{pq}(w)$) and the sum of squared error (E(w)) can be defined as follows:

$$e_{pq}(w) = y_{pq}(w) - Target_{pq}(w) \qquad q \in \{1,2,\dots,M\} \;,\; p \in \{1,2,\dots,N_p\} \tag{14}$$

$$\begin{aligned} E(w) &= \frac{1}{2}\sum_{p=1}^{N_p}\left(e_p(w)\right)^2 = \sum_{p=1}^{N_p}\frac{1}{2}\left(y_{pq}(w) - Target_{pq}(w)\right)^T\left(y_{pq}(w) - Target_{pq}(w)\right) \\ &= \frac{1}{2}\sum_{p=1}^{N_p}\sum_{q=1}^{M}(e_{pq}(w))^2 \end{aligned} \tag{15}$$

Where N_p is the total number of training samples. Therefore, $E(w)$ is the sum of squared residual errors of all network outputs for all training patterns. Finally, the modification of the weights in the Levenberg-Marquardt algorithm is as follows:

$$w(k+1) = w(k) - \eta(k) \times [J^T(k) \times J(k) + \mu(k) \times I]^{-1} \times J^T(k) \times e(k) \quad (16)$$

Where J is the Jacobian matrix and is defined as follows:

$$J = \begin{bmatrix} \frac{\partial e_{11}}{\partial w_1}(k) & \frac{\partial e_{11}}{\partial w_2}(k) & \cdots & \frac{\partial e_{11}}{\partial w_{N_w}}(k) \\ \frac{\partial e_{21}}{\partial w_1}(k) & \frac{\partial e_{21}}{\partial w_2}(k) & \cdots & \frac{\partial e_{21}}{\partial w_{N_w}}(k) \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial e_{N_p 1}}{\partial w_1}(k) & \frac{\partial e_{N_p 1}}{\partial w_2}(k) & \cdots & \frac{\partial e_{N_p 1}}{\partial w_{N_w}}(k) \end{bmatrix} \quad (17)$$

In equation (16), the phrase $\mu(k) \times I$ causes the matrix to move away from the singular condition, which makes the inverse of the phrase inside the bracket can be calculated. When $\mu(k)$ is very large, the term $J^T(k) \times J(k)$ becomes very small compared to $\mu(k) \times I$, which causes the equation (16) to become the equation of the typical gradient descent method.

On the other hand, if $\mu(k)$ is a very small number, then $\mu(k) \times I$ is negligible compared to $J^T(k) \times J(k)$, in which case, the weight modification equation of the Levenberg-Marquardt algorithm becomes the equation of the Gauss-Newton algorithm and subsequently approaches the singularity condition.

2.2.3. Flexible two-layer perceptron

The most important feature of flexible neural networks is that these networks are more similar to the structure of biological neurons in the human brain. According to the need of the problem, some parameters can be considered flexibly in the neural network. Flexible consideration of a parameter increases the flexibility and the degree of freedom of the network. In this research, to create a flexible network, the « g » parameter has been trained.

For the hidden layer, the flexible binary sigmoid (logarithmic sigmoid) activation function is defined as follows:

$$f^1(k) = \frac{2g^1(k)}{1 + e^{-g^1(k) \times Net^1(k)}} \quad (18)$$

The purpose of adjusting the slope coefficients (g) for the activation functions is that the network learns how to update «g» to optimize the cost function. For this purpose, the error backpropagation equations by chain derivatives are used to update the «g» parameter of the activation function.

For the last layer :

$$\Delta g^2(k) = -\eta_g^2 \frac{\partial E(k)}{\partial g^2(k)} = -\eta_g^2 \times \frac{\partial E(k)}{\partial e(k)} \times \frac{\partial e(k)}{\partial O^2(k)} \times \frac{\partial O^2(k)}{\partial g^2(k)} = -\eta_g^2 \times e(k) \times (-1) \times f^{*2}(k) \quad (19)$$

$$\Delta g^2(k) = \eta_g^2 \times e(k) \times f^{*2}(k) = \eta_g^2 \times \sigma^2(k) \quad (20)$$

And, for the hidden layer :

$$\Delta g^1(k) = -\eta_g^1 \frac{\partial E(k)}{\partial g^1(k)} \tag{21}$$

$$\Delta g^1(k) = -\eta_g^1 \times \frac{\partial E(k)}{\partial e(k)} \times \frac{\partial e(k)}{\partial O^2(k)} \times \frac{\partial O^2(k)}{\partial Net^2(k)} \times \frac{\partial Net^2(k)}{\partial O^1(k)} \times \frac{\partial O^1(k)}{\partial g^1(k)} \tag{22}$$

$$\Delta g^1(k) = -\eta_g^1 \times e(k) \times (-1) \times f^{2'}(k) \times w^2(k) \times f^{1^*}(k) \tag{23}$$

$$\Delta g^1(k) = \eta_g^1 \times e(k) \times f^{2'}(k) \times w^2(k) \times f^{1^*}(k) \tag{24}$$

Now, if the network has two activation layers in such a way that the hidden layer has a nonlinear activation function and the output layer has a linear activation function, the previous equations will be expressed more simply as follows:

$$\Delta g^1(k) = -\eta_g^1 \times e(k) \times 1 \times w^2(k) \times f^{1^*}(k) \tag{25}$$

And for g^2, no training is done because the activation function of the output layer is linear, and the g factor is not present in it, so its derivative in terms of g is zero. In other words, $\Delta g^2(k)$ is equal to zero. Mathematically, it can be written:

$$\Delta g^2(k) = \eta_g^2 \times e(k) \times f^{2^*}(k) = \eta_g^2 \times e(k) \times 0 = 0 \rightarrow \Delta g^2(k) = 0 \tag{26}$$

In the above equations, the derivative of the nonlinear activation function of the hidden layer is defined as follows:

$$f^{1^*}(k) = \frac{O^1(k)}{g^1(k)} + \frac{2g^1(k) \times Net^1(k) \times e^{-g^1(k) \times Net^1(k)}}{\left(1 + e^{-g^1(k) \times Net^1(k)}\right)^2} \tag{27}$$

2.2.4. Two-layer perceptron + Rough Model (Type 1) + Gradient Descent

Some types of neural networks can sometimes fail to provide good approximations when the data is polluted with noise. To solve this problem, uncertainty-resistant neural networks have been proposed. One such robust neural network is the rough neural network presented by Lingras [16]. The main difference between these networks and other multilayer neural networks is the backpropagation of neurons and communication weight errors. These networks use interval weights instead of considering deterministic weights to include uncertainty, and use rough neurons instead of deterministic neurons that can be applied to all layers. In a two-layer perceptron, if this operation is applied to the first layer (hidden layer), it is said to be a type 1 rough model.

The symbols U and L represent the upper limit and lower limit neurons, respectively. According to figure 4, the inputs are entered into the upper and lower limit neurons through interval weights. After the effect of the activation function on them, the output O_U for the upper limit neuron and O_L for the lower limit neuron are produced. The final output can then be generated through one of the average or weighted models are shown in figure 4. In this research, the average model is used. It should be noted that the average model is a special case of the

weighted model where $\alpha = \beta = 0.5$. In a more general way, the weighted model can be defined as follows:

$$\beta = 1 - \alpha \quad 0 < \alpha, \beta < 1 \tag{28}$$

$$O = \alpha \times O_U + \beta \times O_L \tag{29}$$

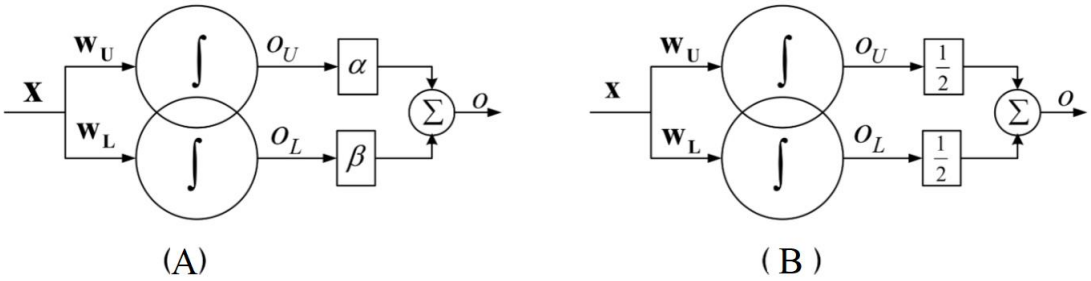


Figure 4.
A: A rough neuron model with weighted output (weighted model)
B: A rough neuron model with average output (average model)

The feedforward equations of the first layer for the type1 rough model with gradient descent learning algorithm are as follows:

$$Net_L^1 = w_L^1 \times Input^T \tag{30}$$

$$Net_U^1 = w_U^1 \times Input^T \tag{31}$$

$$O_L^1 = min\left(f^1(Net_L^1), f^1(Net_U^1)\right) = min\left(\frac{1}{1+exp(-Net_L^1)}, \frac{1}{1+exp(-Net_U^1)}\right) \tag{32}$$

$$O_U^1 = max\left(f^1(Net_L^1), f^1(Net_U^1)\right) = max\left(\frac{1}{1+exp(-Net_L^1)}, \frac{1}{1+exp(-Net_U^1)}\right) \tag{33}$$

$$O^1 = 0.5 \times (O_U^1 + O_L^1) \tag{34}$$

And for the output layer, the feedforward equations are as follows:

$$Net^2 = w^2 \times O^1 \tag{35}$$

$$O^2 = f^2(Net^2) = Net^2 \tag{36}$$

$$e = Target - O^2 \tag{37}$$

Then the steps of backpropagation of the error are performed, that the gradient descent algorithm updates and corrects the weights by propagating the effect of the error from the last layer to the first layer. Thus, for the last layer:

$$\Delta w^2(k) = -\eta^2 \frac{\partial E(k)}{\partial w^2(k)} = -\eta^2 \times \left(\frac{\partial E}{\partial e} \times \frac{\partial e}{\partial O^2} \times \frac{\partial O^2}{\partial Net^2} \times \frac{\partial Net^2}{\partial w^2}\right)(k) \tag{38}$$

$$\Delta w^2(k) = -\eta^2 \times e(k) \times (-1) \times 1 \times {O^1}^T(k) = \eta^2 \times e(k) \times {O^1}^T(k) \tag{39}$$

$$w^2(k+1) = w^2(k) + \Delta w^2(k) = w^2(k) + \eta^2 \times e(k) \times {O^1}^T(k) \tag{40}$$

In backpropagation equations for the hidden layer for type1 rough model with $\alpha = \beta = 0.5$, the sensitivity functions in both modes $f^1(Net_L^1) \le f^1(Net_U^1)$ or $f^1(Net_L^1) > f^1(Net_U^1)$ are the same and have no difference. In fact, it can be said in the backpropagation equations of the first type Rough model that $\alpha = \beta = 0.5$ is used, $f^1(Net_U^1)$ and $f^1(Net_L^1)$ being larger or smaller than each other will not cause sensitivity to the equations of the model, and it will be the same in both case. Therefore, to avoid long explanations, only the first case will be presented.

First case: if for a neuron, $f^1(Net_L^1) \le f^1(Net_U^1)$, Then:

$$O_L^1 = min\left(f^1(Net_L^1), f^1(Net_U^1)\right) = f^1(Net_L^1) = \frac{1}{1 + exp(-Net_L^1)} \tag{41}$$

$$O_U^1 = max\left(f^1(Net_L^1), f^1(Net_U^1)\right) = f^1(Net_U^1) = \frac{1}{1 + exp(-Net_U^1)} \tag{42}$$

$$O^1 = 0.5 \times (O_U^1 + O_L^1) \tag{43}$$

Setting the weights of the lower limit of the hidden layer:

$$\Delta w_L^1 = -\eta^1 \times \left(\frac{\partial E}{\partial e} \times \frac{\partial e}{\partial O^2} \times \frac{\partial O^2}{\partial Net^2} \times \frac{\partial Net^2}{\partial O^1} \times \frac{\partial O^1}{\partial O_L^1} \times \frac{\partial O_L^1}{\partial Net_L^1} \times \frac{\partial Net_L^1}{\partial w_L^1}\right)(k) \tag{44}$$

$$\Delta w_L^1 = -\eta^1 \times e(k) \times (-1) \times 1 \times \left(w^2(k) \times {f^1}'(k)\right)^T \times 0.5 \times Input(k) \tag{45}$$

$${f^1}'(k) = diag\left(O_L^1(k) \times (1 - O_L^1(k))\right) \tag{46}$$

$$\begin{aligned} w_L^1(k+1) &= w_L^1(k) + \Delta w_L^1(k) \\ &= w_L^1(k) + \eta^1 \times e(k) \times \left(w^2(k) \times {f^1}'(k)\right)^T \times 0.5 \times Input(k) \end{aligned} \tag{47}$$

And then, Setting the weights of the upper limit of the hidden layer:

$$\Delta w_U^1 = -\eta^1 \times \left(\frac{\partial E}{\partial e} \times \frac{\partial e}{\partial O^2} \times \frac{\partial O^2}{\partial Net^2} \times \frac{\partial Net^2}{\partial O^1} \times \frac{\partial O^1}{\partial O_U^1} \times \frac{\partial O_U^1}{\partial Net_U^1} \times \frac{\partial Net_U^1}{\partial w_U^1}\right)(k) \tag{48}$$

$$\Delta w_U^1(k) = -\eta^1 \times e(k) \times (-1) \times 1 \times \left(w^2(k) \times {f^1}'(k)\right)^T \times 0.5 \times Input(k) \tag{49}$$

$${f^1}'(k) = diag\left(O_U^1(k) \times (1 - O_U^1(k))\right) \tag{50}$$

$$\begin{aligned} w_U^1(k+1) &= w_U^1(k) + \Delta w_U^1(k) \\ &= w_U^1(k) + \eta^1 \times e(k) \times \left(w^2(k) \times {f^1}'(k)\right)^T \times 0.5 \times Input(k) \end{aligned} \tag{51}$$

Figure 5 shows a generalized form of the training process of a two-layer perceptron neural network based on the first type of rough model.

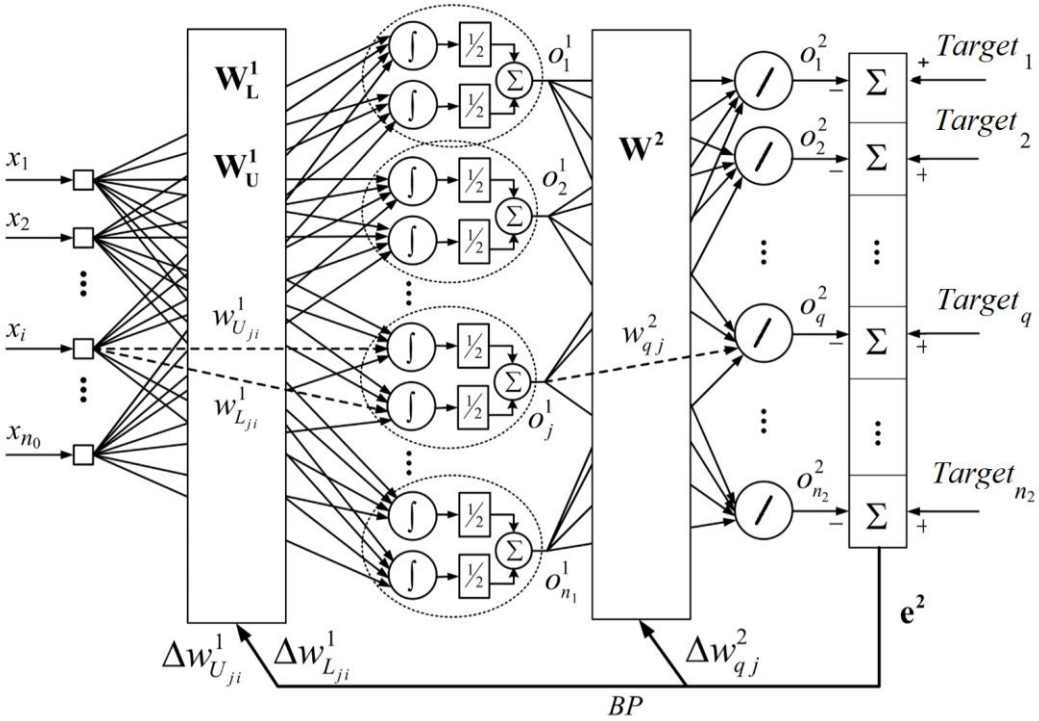


Figure 5. A two-layer Rough (type1) perceptron neural network training process.

2.2.5. Two layer perceptron + Recurrent Neural Network (Elman) + Gradient Descent

Elman introduced a different neural network with internal feedback in the hidden layer [17]. This neural network is one of the simplest recurrent neural networks whose parameters can be adjusted based on the error backpropagation method. If the return weights are not adjustable, the Elman network is partially recurrent, and if the return and forward weights are adjustable, the Elman network is fully recurrent. In this research, a fully recurrent network has been implemented.

The feedforward equations for the Elman neural network can be written as follows:

$$Net^1(k) = w_x(k) \times Input(k) + w_c(k) \times x_c(k) \tag{52}$$

$$o^1(k) = f^1\big(Net^1(k)\big) = f^1(w_x(k) \times Input(k) + w_c(k) \times o^1(k-1)) \tag{53}$$

In the above equations, Input=$[x_1, x_2, \ldots, x_{n_0}]$ is the external input vector, and x_c is the output of the hidden layer in the previous time step that is mathematically expressed as $x_c(k) = o^1(k-1)$. Also, $f^1$is the logarithmic sigmoid activation function (for the hidden layer). w_c and $w_x$are the recurrent and forward weight matrices, respectively. Therefore, the output of the hidden layer activation function depends on the information about its previous step, and this indicates the depth of the network memory.

The feedforward equations for the output layer are:

$$Net^2(k) = w_y(k) \times o^1(k) \quad (54)$$
$$o^2(k) = f^2\big(Net^2(k)\big) \quad (55)$$

That w_y is the forward weight matrix between the hidden layer and output layer. o^2 is the final output vector of the neural network. The activation function of the output layer is also considered linear. Therefore:

$$o^2(k) = Net^2(k) \quad (56)$$

Gradient descent error backpropagation algorithm is used to train network weights. The function that shows the performance of the network during the training process is:

$$E(k) = \frac{1}{2}e^2(k) = \frac{1}{2}(Target(k) - o^2(k))^2 \quad (57)$$

The training of the weights between the output and hidden layer by the gradient descent method is as follows:

$$\frac{\partial E(k)}{\partial w_y(k)} = \frac{\partial E(k)}{\partial e(k)} \times \frac{\partial e(k)}{\partial o^2(k)} \times \frac{\partial o^2(k)}{\partial Net^2(k)} \times \frac{\partial Net^2(k)}{\partial w_y(k)} = e(k) \times (-1) \times f^{2'}(k) \times o^1(k) \quad (58)$$
$$\Delta w_y(k) = -\eta_y \frac{\partial E(k)}{\partial w_y(k)} = \eta_y \times e(k) \times f^{2'}(k) \times o^1(k) \quad (59)$$
$$w_y(k+1) = w_y(k) + \Delta w_y(k) = w_y(k) + \eta_y \times e(k) \times f^{2'}(k) \times o^1(k) \quad (60)$$

Considering that according to the above equation, the information of the previous step is present in $o^1(k)$, so the memory depth has also appeared in the above equation. The steps of adjusting the forward weight matrix between the hidden layer and the input layer, is done as follows:

$$\frac{\partial E(k)}{\partial w_x(k)} = \frac{\partial E(k)}{\partial e(k)} \times \frac{\partial e(k)}{\partial o^2(k)} \times \frac{\partial o^2(k)}{\partial Net^2(k)} \times \frac{\partial Net^2(k)}{\partial o^1(k)} \times \frac{\partial o^1(k)}{\partial Net^1(k)} \times \frac{\partial Net^1(k)}{\partial w_x(k)} \quad (61)$$
$$\Delta w_x(k) = e(k) \times (-1) \times f^{2'}(k) \times w_y(k) \times f^{1'}(k) \times Input(k) \quad (62)$$
$$\Delta w_x(k) = -\eta_x \frac{\partial E(k)}{\partial w_x(k)} = \eta_x \times e(k) \times f^{2'}(k) \times w_y(k) \times f^{1'}(k) \times Input(k) \quad (63)$$
$$w_x(k+1) = w_x(k) - \eta_x \frac{\partial E(k)}{\partial w_x(k)} = w_x(k) + \eta_x \times e(k) \times f^{2'}(k) \times w^2(k) \times f^{1'}(k) \times Input(k) \quad (64)$$

In the following, the weights of the recurrent sections are also obtained in the same way by the gradient descent algorithm:

$$\frac{\partial E(k)}{\partial w_c(k)} = \frac{\partial E(k)}{\partial e(k)} \times \frac{\partial e(k)}{\partial o^2(k)} \times \frac{\partial o^2(k)}{\partial Net^2(k)} \times \frac{\partial Net^2(k)}{\partial o^1(k)} \times \frac{\partial o^1(k)}{\partial Net^1(k)} \times \frac{\partial Net^1(k)}{\partial w_c(k)} \quad (65)$$

$$\Delta w_c(k) = -\eta_c \frac{\partial E(k)}{\partial w_c(k)} \tag{66}$$

The chain dependency between the output of the hidden layer and the return weights means that $o^1(k-1)$ depends on $w_c(k-1) \times o^1(k-2)$. Also, $o^1(k-2)$ depends on $w_c(k-2) \times o^1(k-3)$ and etc. This chain dependence is called backpropagation through time in the calculation of chain derivatives.

$$\Delta w_c(k) = \eta_c \times e(k) \times f^{2'}(k) \times w_y(k) \times f^{1'}(k) \times \left[x_c(k) + w_c(k) \times \frac{\partial x_c(k)}{\partial w_c(k)}\right] \tag{67}$$

Or:

$$\Delta w_c(k) = \eta_c \times e(k) \times f^{2'}(k) \times w_y(k) \times f^{1'}(k) \times \left[x_c(k) + w_c(k) \times \frac{\partial o^1(k-1)}{\partial w_c(k)}\right] \tag{68}$$

If the dependence between o^1 and w_c is ignored from a certain time before, $\frac{\partial o^1(k-1)}{\partial w_c(k)}$ can be considered equal to zero. It is called truncated backpropagation through time. In this research, truncated backpropagation through time was used. Therefore, the above equation is expressed in a simplified form as follows:

$$\Delta w_c(k) = \eta_c \times e(k) \times f^{2'}(k) \times w_y(k) \times f^{1'}(k) \times x_c(k) \tag{69}$$

$$w_c(k+1) = w_c(k) + \Delta w_c(k) = w_c(k) + \eta_c \times e(k) \times f^{2'}(k) \times w_y(k) \times f^{1'}(k) \times x_c(k) \tag{70}$$

2.3. Accuracy and error evaluation criteria

In this study, several functions have been used to evaluate the accuracy of the model, which include:

MSE, RMSE, MAE and R^2. In the continuation of this section, the formulation of each of these four criteria will be explained:

2.3.1. MSE (Mean Squared Error):

The MSE index is the mean square of errors and is defined mathematically as the following equation:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{71}$$

That N is the number of data, y_i is the i-th target, and $\hat{y}_i$ is the i-th predicted data. As MSE gets closer to zero, the quality of the results improves.

2.3.2. RMSE (Root Mean Square Error):

The RMSE index is the root mean square of errors and is mathematically defined as the following equation:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{72}$$

In other words, RMSE is equivalent to the square root of MSE.

$$RMSE = \sqrt{MSE} \tag{73}$$

As $RMSE$ gets closer to zero, the quality of the results improves.

2.3.3. MAE (Mean Absolute Error):

In statistics, mean absolute error is a criterion of errors between paired observations expressing the same phenomenon. This index means the average of the absolute values of the errors and is expressed mathematically as the following equation:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{74}$$

As MAE gets closer to zero, the quality of the results improves.

2.3.4. R^2 (Coefficient of Determination):

The coefficient of determination, represented by the symbol R^2, expresses the proportion of the variation in the dependent variable that is predictable from the independent variable. It is expressed mathematically as the following equation:

$$R^2 = 1 - \frac{RSS}{TSS} \tag{75}$$

$$RSS = \sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{76}$$

$$TSS = (N-1) \times Variance = (N-1) \times \sum_{i=1}^{N}\frac{(y_i - \overline{y_i})^2}{N-1} = \sum_{i=1}^{N}(y_i - \overline{y_i})^2 \tag{77}$$

RSS means the sum of squares of residuals, TSS means the total sum of squares, and $\overline{y_i}$ is the mean of data. As R^2 gets closer to 1, the quality of the results improves.

3. Results

The results implementing the five mentioned models are presented in Table 3. Error indices show that the Levenberg-Marquardt method is more efficient than the other methods and provides a more accurate prediction. After that, the rough model is in second place and provides a more accurate prediction than the other methods.

Table 3. Error indices of MLP during the training and testing stage

Model	$R^2{}_{Train}$	$R^2{}_{Test}$	$RMSE_{Train}$	$RMSE_{Test}$	MSE_{Train}	MSE_{Test}	MAE_{Train}	MAE_{Test}
1	0.83696	0.8442	0.014575	0.011946	0.000212	0.000142	0.011178	0.009455
2	0.92616	0.94079	0.009808	0.007364	0.000096	0.000054	0.007815	0.005801
3	0.84946	0.84552	0.014005	0.011895	0.000196	0.000141	0.010940	0.009131
4	0.91598	0.92205	0.010463	0.008449	0.000109	0.000071	0.008401	0.006305
5	0.86330	0.84206	0.013346	0.012028	0.000178	0.000144	0.010586	0.008778

According to the results presented in table 3, the second model (Levenberg Marquardt algorithm) and the fourth model (Rough Model) were determined as the chosen model of this research. The results of the development process of the second model (Levenberg Marquardt algorithm) during the training and the test stage are shown in figures (6) and (7), respectively:

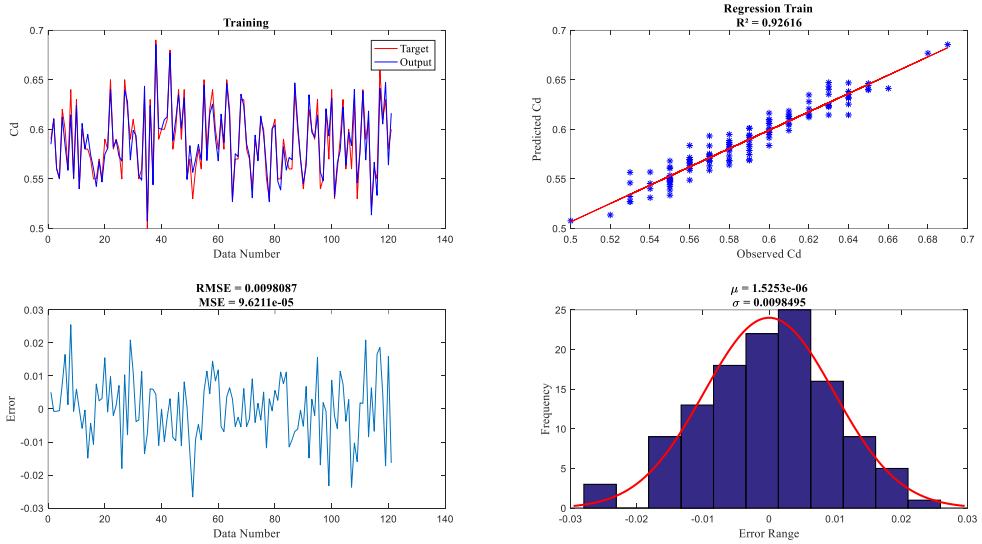


Figure 6. ANN development process during the training stage (Levenberg-Marquardt Model)

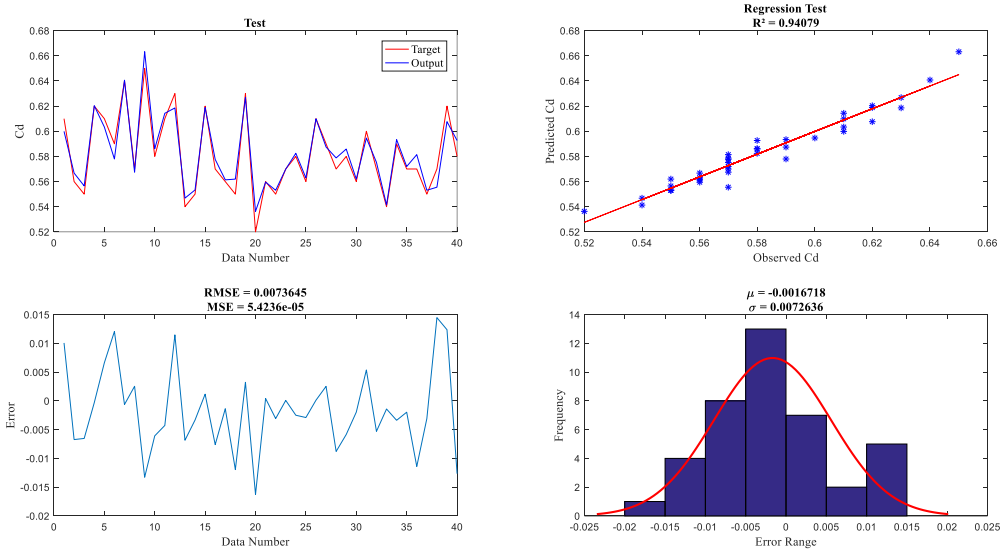
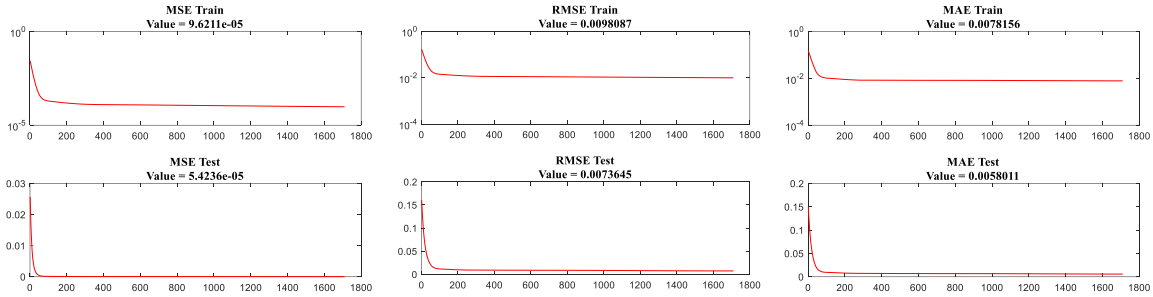


Figure 7. ANN development process during the test stage (Levenberg-Marquardt Model)

Figure (8) also shows how error indicators such as MSE, RMSE, and MAE are developed during the training and test stage process and reach a stable level.



MSE RMSE MAE

Figure 8. Error indicators development process during training & test stage (Levenberg-Marquardt Model)

Then the results of the development process of the fourth model (Rough model) during the training and the test stage are shown in figures (9) and (10), respectively:

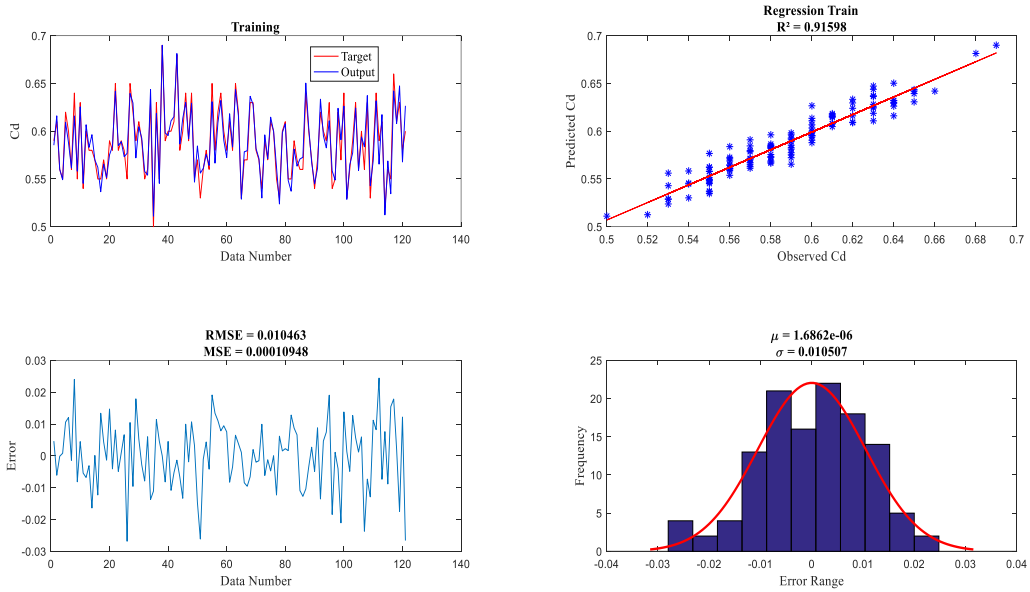


Figure 9. ANN development process during the training stage (Rough Model)

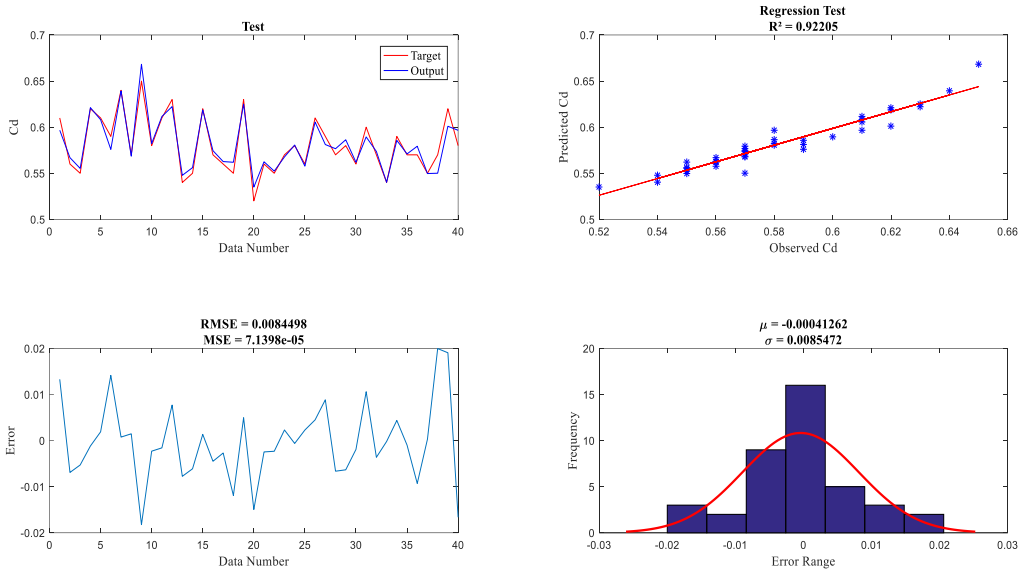
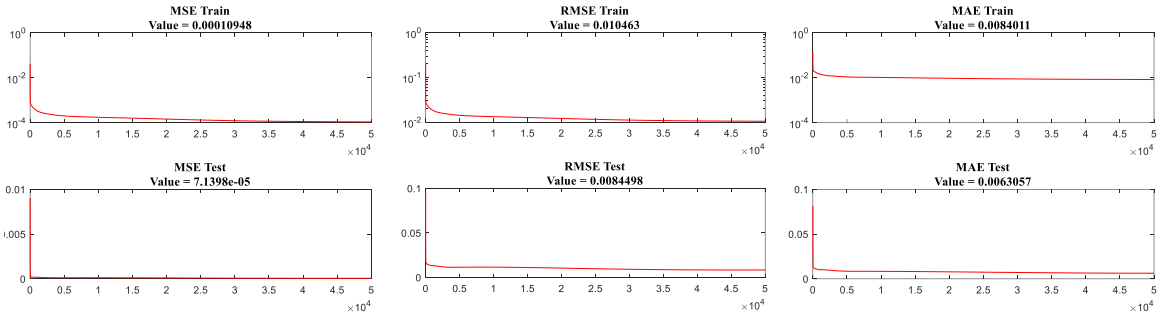


Figure 10. ANN development process during the test stage (Rough Model)

Figure (11) also shows how error indicators such as MSE, RMSE, and MAE are developed during the training and test stage process and reach a stable level.



MSE RMSE MAE

Figure 11. Error indicators development process during the training & the test stage (Rough Model)

4. Discussion

In Table 4, a comparison of the results between the current research and the previous research is presented. This comparison shows that the results were obtained from the Levenberg-Marquardt algorithm in this research are better than the research of Parsaie et al. [5].

Table 4. A comparison between the current research and Parsaie et al. [5]

Model	N.I	N.H.L	A.F.1	N.N.H.L	R^2_{Train}	R^2_{Test}	$RMSE_{Train}$	$RMSE_{Test}$
LM (This research)	4	1	logsig	7	0.92616	0.94079	0.009808	0.007364
LM (Parsaie et. al.)	4	1	logsig	6	0.88	0.84	0.012	0.014

Notes: LM: Levenberg-Marquardt, N.I: number of inputs, N.H.L: number of hidden layers, N.N.H.L: number of neurons in the hidden layer, A.F.1: activation function of hidden layer

5. Sensitivity Analysis

Determining the effective parameters for each problem is very important in soft computing. Therefore, in this section, the importance of each parameter in the model's results is examined. For this purpose, various approaches can be considered. In this research, the influence of inputs on the model's performance has been investigated. In this way, every time, by removing one of the four input parameters, the model is trained using the remaining three input parameters. Therefore, by examining the error indicators in each situation, it is determined which of the parameters the model is more sensitive to than other parameters. The results of the sensitivity analysis are presented in Table (5).

Table 5. Sensitivity Analysis of ANN

Input Items	Absent Item	Target	R^2_{Test}	$RMSE_{Test}$	MSE_{Test}	MAE_{Test}
$y/d, b/d, b/B, H/d$	–	C_d	0.94079	0.0073645	0.000054	0.005801
$b/d, b/B, H/d$	y/d	C_d	0.86952	0.010932	0.000119	0.008533
$y/d, b/B, H/d$	b/d	C_d	0.92073	0.008520	0.000072	0.006746
$y/d, b/d, H/d$	b/B	C_d	0.89681	0.009722	0.000094	0.007460
$y/d, b/d, b/B$	H/d	C_d	0.74	0.023463	0.000551	0.017498

Based on the sensitivity analysis, it can be seen that the model has minimum sensitivity to the dimensionless parameter b/d. Also, the highest sensitivity of the model results is related to the dimensionless parameter H/d.

6. Conclusions

In this research, five different methods based on artificial neural networks were used to predict the discharge coefficient of combined weir-gate. The results (presented in Table 3) showed that the model trained by the Levenberg-Marquardt algorithm shows a more accurate prediction than the other models. Also, in another part of this study, sensitivity analysis was performed for all four input parameters of this model. Based on the results presented in table (4), the highest sensitivity of the model results is related to the H/d parameter. Also, the model shows minimum sensitivity to the b/d parameter compared to others.

References

1. Nouri, M., & Hemmati, M. (2020). Discharge coefficient in the combined weir-gate structure. Flow Measurement and Instrumentation, 75, 101780.
2. Chen, W., Sharifrazi, D., Liang, G., Band, S. S., Chau, K. W., & Mosavi, A. (2022). Accurate discharge coefficient prediction of streamlined weirs by coupling linear regression and deep convolutional gated recurrent unit. Engineering Applications of Computational Fluid Mechanics, 16(1), 965-976.
3. Safari, S., Takarli, A., Salarian, M., Banejad, H., Heydari, M., & Ghadim, H. B. (2022). Evaluation of ANN, GEP, and Regression Models to Estimate the Discharge Coefficient for the Rectangular Broad-Crested Weir. Polish Journal of Environmental Studies, 31(5).
4. Sahib, J. H., Al-Waeli, L. K., & Al Rammahi, A. H. J. (2022). Utilization of ANN technique to estimate the discharge coefficient for trapezoidal weir-gate. Open Engineering, 12(1), 142-150.

5. Parsaie, A., Haghiabi, A. H., Emamgholizadeh, S., & Azamathulla, H. M. (2019). Prediction of discharge coefficient of combined weir-gate using ANN, ANFIS and SVM. International Journal of Hydrology Science and Technology, 9(4), 412-430.
6. Aein, N., Najarchi, M., Mirhosseini Hezaveh, S. M., Najafizadeh, M. M., & Zeighami, E. (2020, October). Simulation and prediction of discharge coefficient of combined weir–gate structure. In Proceedings of the Institution of Civil Engineers-Water Management (Vol. 173, No. 5, pp. 238-248). Thomas Telford Ltd.
7. Ahmed, F. H. (1985). Characteristics of discharge of the combined flow through sluice gates and over weirs. J. Engineering and Technology, Iraq, 3(2), 49-63.
8. Negm, A. M., El-Saiad, A. A., Alhamid, A. A., & Husain, D. (1995). Characteristics of simultaneous flow over weir and below inverted V-Notches. Civil Engineering Research Magazine, Civil Engineering Department, Faculty of Engineering, Al-Azhar University, Cairo, Egypt, 16(19), 786-799.
9. Hayawi, H. A. A. M., Yahia, A. A. A. G., & Hayawi, G. A. M. (2008). Free combined flow over a triangular weir and under rectangular gate. Damascus university journal, 24(1), 9-22.
10. Wang, F., Zheng, S., Ren, Y., Liu, W., & Wu, C. (2022). Application of hybrid neural network in discharge coefficient prediction of triangular labyrinth weir. Flow Measurement and Instrumentation, 83, 102108.
11. Bilhan, O., Emiroglu, M. E., & Kisi, O. (2011). Use of artificial neural networks for prediction of discharge coefficient of triangular labyrinth side weir in curved channels. Advances in Engineering Software, 42(4), 208-214.
12. Salmasi, F., Nahrain, F., Abraham, J., & Taheri Aghdam, A. (2023). Prediction of discharge coefficients for broad-crested weirs using expert systems. ISH Journal of Hydraulic Engineering, 29(1), 1-11.
13. Parsaie, A., Haghiabi, A. H., Saneie, M., & Torabi, H. (2017). Predication of discharge coefficient of cylindrical weir-gate using adaptive neuro fuzzy inference systems (ANFIS). Frontiers of Structural and Civil Engineering, 11(1), 111-122.
14. Lucas C., Abbaspour A., Gholipour A., Araabi B., Fatourechi M. (2003). "Enhancing the performance of neurofuzzy predictors by emotional learning algorithm", Informatica, 27, 137–145.
15. Roweis, S. (1996). Levenberg-marquardt optimization. Notes, University Of Toronto, 52.
16. Lingras, P. (1996, July). Rough neural networks. In Proc. of the 6th Int. Conf. on Information Processing and Management of Uncertainty in Knowledgebased Systems (pp. 1445-1450).
17. Elman, J.L. (1990). Finding structure in time. Cognitive Science, 14, 179-211.