Mars: Jurnal Teknik Mesin, Industri, Elektro dan Ilmu Komputer
Volume. 3 Nomor. 1 Tahun 2025
e-ISSN : 3031-8742, dan p-ISSN : 3031-8750, Hal. 194-206
DOI: https://doi.org/10.61132/mars.v3i1.666
Available online at: https://journal.arteii.or.id/index.php/Mars

Application of Neural Networks in Prediction of Software Effort

Zainab Rustum Mohsin
Department of Information Technology, College of Computer Science and Mathematics, University of Thi-Qar, Thi-Qar, Iraq.

Author correspondence: zainabrustum@utq.edu.iq

Abstract. *Estimating the effort, time, and cost needed to build a software project is an important task in software engineering. Estimating software prior to development can help to reduce risk and improve the project success rate. Researchers have developed numerous traditional and machine learning models to estimate software effort, but it has always been difficult to estimate effort precisely. This paper presents a predictive model based on artificial neural networks namely ANNs to predict the software effort. The NASA dataset is applied to construct the proposed model. The system was trained using 50 data points, and the remaining 10 were used for testing. It was concluded that the ANN approach could estimate the software effort with high accuracy. A comparative study with other published equations was also performed, and it was found that ANN had less error and produced better results than other existing methods.*

Keywords*: ANN, Machine Learning, NASA, Software Effort*

1. INTRODUCTION

A highly difficult challenge task for project managers is estimating the cost and effort required in the early stages of planning. Software projects necessitate effective planning to ensure on-time delivery, within budget, and to required quality standards. Underestimation the effort can cause a delay and cost over-run, causing project failure, as well as its overestimation the effort may leading to a financial loss for a company [1]. Accordingly, the success or failure of any software project it seems to be heavily dependent on how accurate its effort estimates are [2]. To do this, a reliable approaches should be available to conduct project feasibility analysis, project planning, allocation of resource, overall project cost estimation and project scheduling [3]. Soft computing (SC) approaches like artificial neural network (ANN) have recently been used widely as a flexible tool for modelling the complex relationship between software attributes and effort. The advantages of adopting SC techniques over other methods because their capability to self-learn from the data and, as a result, reduce error. SC involves some effective techniques for solving complex problems using the human tolerance for partial truth, imprecision, uncertainty, and approximation [4].

By using these methods, a significant amount of time and cost would be saved while also improving accuracy. In this paper, the ability of one of the most commonly approaches in soft computing namely artificial neural networks (ANNs) is employed to estimate the software effort. The NASA database is applied as the dataset to train and test the presented ANN model.

Received November 17,2024; Revised Desember 19,2024; Accepted Januari 20, 2025; Online Available: Januari 23, 2025

The obtained results of the proposed model were also compared with other existing equations, and good agreement is achieved.

The organization of this paper is as follows: in section 2, the systematic literature review that was conducted are described briefly. Section 3 describes the ANN technique Section 4 shows the evaluation criteria used. Section 5 gives detail about the datasets and the structure of the proposed model used in this paper. Experimental results, comparison, and conclusion are presented in Section 6, Section 7, and Section 8 respectively.

2. LITERATURE REVIEW

Related Work

ANN technique is widely used in software effort estimation because of their ability to learn and generalize from data . Numerous researchers used various ANN methods and various datasets to predict the effort more precisely.

Rijwani and Jain [5] used back-propagation ANN to estimate software effort. The predictive model has been constructed by using COCOMO dataset. Mohsin [6] implemented a back-propagation ANN for effort estimation a. The COCOMO datasets were used for training and testing the model. Jaswinder Kaur, et al [7] modelled the software effort utilizing a back propagation ANN technique . The authors used NASA dataset which consist of 18 projects to build the proposed model. Roheet Bhatnagar, et al.[8] employed the ANN approach to model the software development effort. They used the dataset supplied by Lopez-Martin, which contains data from 41 project. K.K. Aggarwal , et al.[9] used various training algorithms for ANN to find which is the best suited for using in effort estimation. ISBSG repository data was used to develop the ANN model. It has been observed that the ‘trainbr’ is the best algorithm. G. E. Wittig, et al.[10] investigated the ability of ANN technique in predicting the software development effort. They utilized a datasets of 15 commercial systems. Kumar et al. [11] explore the ability of wavelet neural network (WNN) for computing the software effort estimation. Two sets of data have been employed : the IBM data processing services, which included 37 projects, and the Canadian financial, which included 24 projects. The proposed WNN demonstrated more accuracy when compared to the other models. Nassif et al. [12] conducted a study for predicting the software development effort by using the ISBSG datasets. Four ANNs techniques (multi layer perceptron effort, general regression NNs, cascade correlation NNs, and radial basis function NNs) were used. Their results showed that the cascade correlation NNs performed better than the other techniques. Finally, in recent years, great interest in using soft computing techniques especially ANNs has increased. The ANN method has been used successfully to a variety of problems. They can be employed as predictive models since they develop methods for modeling complex and non-linear ANN technique is widely used in software effort estimation because of its ability to learn and generalize from data. Numerous researchers used various ANN methods and various datasets to predict the effort more precisely.

Rijwani and Jain [5] used back-propagation ANN to estimate software effort. The predictive model has been constructed by using COCOMO dataset. Mohsin [6] implemented a back-propagation ANN for effort estimation a. The COCOMO datasets were used for training and testing the model. Jaswinder Kaur, et al [7] modelled the software effort utilizing a back propagation ANN technique . The authors used NASA dataset which consist of 18 projects to build the proposed model. Roheet Bhatnagar, et al.[8] employed the ANN approach to model the software development effort. They used the dataset supplied by Lopez-Martin, which contains data from 41 projects. K.K. Aggarwal, et al.[9] used various training algorithms for ANN to find which is the best suited for use in effort estimation. ISBSG repository data was used to develop the ANN model. It has been observed that the 'trainbr' is the best algorithm. G. E. Wittig, et al.[10] investigated the ability of ANN technique to predict the software development effort. They utilized a dataset of 15 commercial systems. Kumar et al. [11] explore the ability of wavelet neural network (WNN) for computing the software effort estimation projects, and the Canadian financial, which included 24 projects. The proposed WNN demonstrated more accuracy when compared to the other models. Nassif et al. [12] conducted a study to predict the software development effort by using the ISBSG datasets. Four ANN techniques (multi-layer perceptron effort, general regression NNs, cascade correlation NNs, and radial basis function NNs) were used. Their results showed that the cascade correlation NNs performed better than the other techniques. Finally, in recent years, great interest in using soft computing techniques especially ANNs has increased. The ANN method has been used successfully to a variety of problems. They can be employed as predictive models since they develop methods for modeling complex and non-linear relationships.

Artificial Neural Network Model

Artificial neural networks (ANN) defines as a mathematical model that simulates the human brain processes. weight, summing their product, and then utilizing the activation function to generate the desired output. If the sum exceeds a certain value known as the threshold, then this output can be either excitatory (positive) or inhibitory (negative) input to neuron in the network. This process continuous It consists of a large number of simple processing units or nodes called artificial neurons [13]. Information is flowed through the network along interconnections. At each node, an input value is multiplied by its corresponding till final outputs are obtained [14]. Figure 1 depicts the structure of a single artificial neuron which includes inputs parameter, weights, transfer functions, activation functions, threshold and outputs.

The neural network are fed by numbers of inputs data. As an input enters the node, it is multiplied by its associated weight value. Weight typically represents the strength of the connection between nodes within a network. All weighted inputs are summed up within the artificial neurons and then the non-linear transfer function is utilized to produce the desired output [4], as provided in Equation (1):

$$\text{Total activation: } net\,j = \sum_{i=1}^{n}\left(w_{ij}\right)x_i - \mathrm{T}_j \qquad (1)$$

Where j is the neuron number, n is the number of nodes, w_{ij} is connection weight value, x_i is input value, and T_j is known as the bias of the neuron and is equivalent to the neuron's negative threshold value.

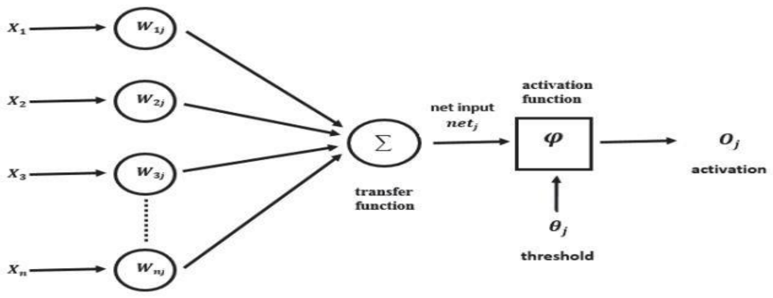

Figure 1. Typical architecture of artificial neurons [15].

The feed-forward backpropagation algorithm is used in the considered ANN simulations. An ANN structure contains three layers: an input layer which presents input parameters to the network, one or more hidden layers, and the output layer which involves a single neuron representing target variables number.

The trials indicated that the two-hidden-layer network outperformed the one-hidden-layer network. A trial and error method was used to determine the optimal number of nodes in the hidden layers. The Back-Propagation Neural Network (BPNN) is trained by feeding a set of input-output variables. The main goal of the training process is to modify the connection weights to an acceptable level by reducing the errors between the target output and the predicted output of the ANN. To get the optimum model performance, the number of hidden layers, numbers of hidden neurons, transfer functions, and data normalization are all chosen. Once errors have been reduced to a minimum, testing is performed using a different set of data that was not utilized in the training phase to determine the efficacy of the proposed ANN model. This process is referred to as generalization of the network. There is no additional weight adjustment during this process. Details of the architecture and procedures of ANN are explained in the available literature [16].

Evaluation Criteria for Model Performance

Mean Magnitude Relative Error (MMRE)

The most commonly used measure for evaluating the efficiency of estimated effort model is MMRE. It gives the average of the difference between the estimated values by proposed model and the actual values for all the projects. MMRE can be calculated using equation 2.

$$MMRE = \frac{1}{N}\sum_{1}^{N}\frac{|Actual\ Effort - Predicted\ Effort|}{Actual\ Effort} \qquad (2)$$

The Correlation Coefficient (R):

The Correlation coefficient (R) is the most common way to measure a linear association between two variables. It is a number **between -1** and +1.

When the **R-value** is closer to 1, it indicates a strong linear relationship. A negative R-value indicates that there is no correlation between the data and the model. The value of R can be found using equation 3:

$$R = 1 - \sqrt{\frac{\sum_{i=1}^{N}(Actual\ Effort - Predicted\ Effort)^2}{\sum_{i=1}^{N}(Actual\ Effort)^2}} \qquad (3)$$

The Root Mean Square Error (RMSE):

The root mean square error (RMSE) is a metric commonly used to evaluate the accuracy of predictions.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(\text{Actual Effort} - \text{Predicted Effort})^2}{N}} \qquad (4)$$

Where N is the total number of software projects.

Experiment

Data Description

NASA data projects collection for the determining of development effort with the ANN model has been used effort using in the present study. This dataset is comprised of 60 projects data. There were 17 attributes in this dataset, however only four attributes were taken into consideration, namely CPLX, PCAP, and KLOC. as shown in Table 1.

The variables that are used to predict other variables are known as independent variables, whereas the variables that are estimated are known as the dependent variable. In this paper, the independent variables are PCAP (programmer capability), CPLX (product complexity), and KLOC (thousand of lines code). While, the dependent variable was the development effort. Since the CPLX and PCAP values were in fuzzy format, we had to convert them to numeric format as follows for the experiment:

a. Very high.
b. High.
c. Normal.
d. extra high.
e. low.

The database was split into two groups including the 50 and 10 projects to train and test the proposed ANN model.

Table 1: NASA Dataset.

Project No.	PCAP	CPLX	KLOC	Actual Effort
1	1	1	70	278
2	2	2	177.9	1248
3	1	2	227	1181
4	3	2	115.8	480
5	3	2	19.7	60
6	3	2	29.5	120
7	3	2	66.6	300
8	3	2	10.4	50
9	3	2	5.5	18
10	3	2	14	60
11	3	2	6.5	42
12	3	2	16	114
13	3	2	13	60
14	3	2	8	42
15	3	2	15	90
16	2	2	90	450
17	2	2	38	210
18	3	2	32.6	170
19	3	2	12.8	62
20	3	2	15.4	70
21	2	3	10	48
22	3	2	16.3	82
23	3	2	161.1	815
24	3	2	48.5	239
25	3	2	35.5	192
26	3	2	7.7	31.2
27	3	2	25.9	117.6
28	3	2	24.6	117.6
29	3	2	9.7	25.2
30	3	2	3.5	10.8
31	3	2	2.2	8.4
32	3	2	8.2	36
33	1	2	150	324
34	3	2	66.6	352.8
35	3	2	100	360
36	1	2	100	360
37	2	2	100	215
38	2	2	15	48

39	2	2	31.5	60
40	3	2	32.5	60
41	3	2	11.3	36
42	2	2	6	24
43	2	2	20	48
44	1	2	20	72
45	2	2	302	2400
46	2	2	7.5	72
47	3	2	219	2120
48	3	2	370	3240
49	2	2	101	750
50	3	2	50	370
51	1	3	190	420
52	3	2	47.5	252
53	3	4	21	107
54	1	3	423	2300
55	2	3	79	400
56	3	5	284.7	973
57	3	5	282.1	1368
58	2	2	78	571.4
59	2	2	11.4	98.8
60	2	2	19.3	155

Generation of the ANN Model

The NN toolbox [17] provided in MATLAB program was used to build the ANN predictive model. A feed-forward propagation algorithm was utilized for modelling the ANN. The sigmoid transfer function was used in the hidden layers and the linear transfer function was applied for the output layer. By using the trial-and-error method, the best epochs, hidden layer numbers, and hidden layer node numbers were found. The training convergence of the ANN model is based on minimizing the for root mean square error (RMSE) error tolerance throughout the training phase and comparing the outputs to evaluate the performance of the ANN model. After the errors have been reduced, testing is done to determine whether the predicted results are reasonably close to the target data. Every neural network consists of an input layer with units representing the input data and an output layer with single node provides development effort. Two hidden layers are also included as intermediate layers. The trials revealed that the network with two-hidden layer outperforms the network with one-hidden layer. The optimum number of nodes in the hidden layers giving the best ANN structure was determine to be 10 for the first hidden layer and 9 for second hidden layer. Therefore, the optimal ANN model was determined to be ANN (3-10-9-1). The architecture of neural network model is presented in Figure 2. The ANN model was continuously trained using update weights until it attained a final error of 0.0066 after 27 epochs. The performance of the proposed ANN model for the testing dataset are presented in Table 2.

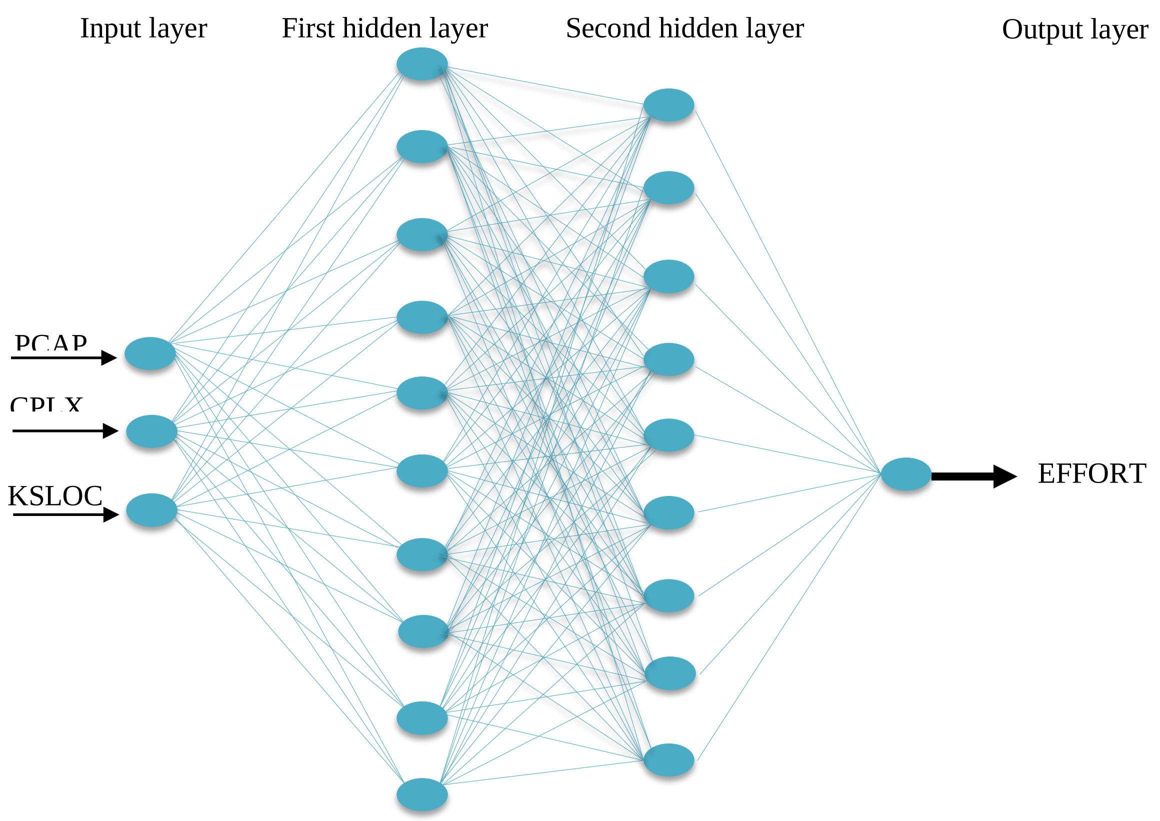

Figure 2: The structure of the proposed ANN model used in this paper.

Table 1. Effort estimation using ANN technique

Project ID	Actual effort	predicted effort with ANN
51	420	524.23
52	252	213.89
53	107	229.89
54	2300	2648.24
55	400	313.1
56	973	1223.32
57	1368	1140.28
58	571.4	359.4
59	98.8	96.46
60	155	115.27

3. RESULTS AND DISSECTIONS

The predictive capability of the proposed ANN model was evaluated using the testing datasets, as shown in Figure 3. As it can be observed from the figure, the predictive values match well with the actual values. The values of R was 0.976 for testing phase. Figure 4 depicts a comparison between actual effort with results estimated by the ANN model for testing set. There was a good agreement between both data sets. Table 3 summarizes the performance of proposed model for testing dataset in term of MMRE, R, and RMSE. According to Table 3, the value of MMRE and RMSE is in the acceptable rang. The ANN model gives a correlation coefficient R for testing data sets of 0.976 which are very close to unity. The results revealed that the ANN approach was able to be effective in software effort estimation.

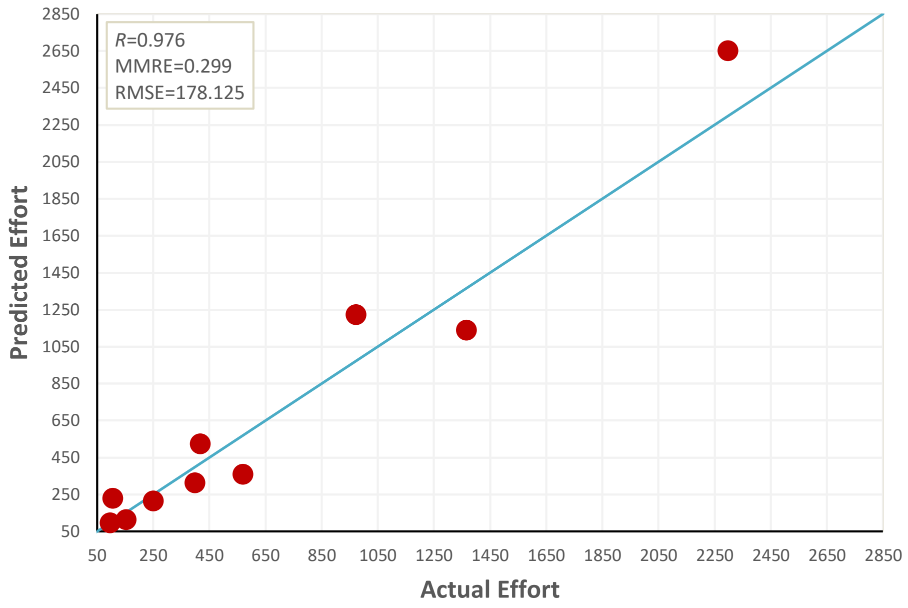

Figure 3. The results of the proposed ANN vs. actual data for the testing set.

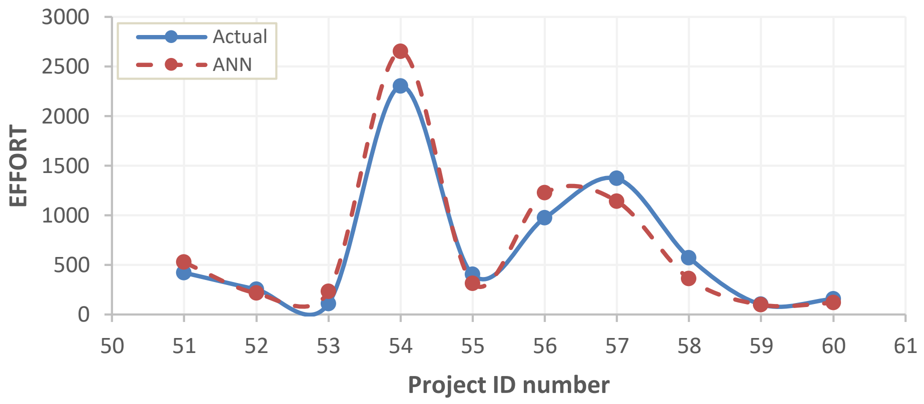

Figure 4. A head to head comparison of performance for proposed ANN model and actual data for testing set.

Table 3. Results of the ANN model

Performance criteria	Predicted effort with ANN
MMRE	0.299
R	0.976
RMSE	178.13

Comparison with Existing Methods

The ability of the proposed ANN model to predict the software development effort was compared to the outputs of the four existing models such as the Walston - Felix model, Halstead model, Doty model, and Bailey-Basili model. Table 4 presents a summary of the considered equations. The results of the models including the ANN model for the testing data sets are plotted against the actual data in Figures 5 and 6. As demonstrated in these figures, the values of the proposed ANN match well with the actual values and perform better compared with the other previous models. For a better comparison, the results of the criteria RMSE and MMRE for all models are given in Table 5. In order to select the model with the best performance, the RMSE and MMRE computed for test data must have the lowest value. According to Table 5, the proposed ANN has the lowest values of RMSE and MMRE. These values indicate that the model presented in this study has best performance in predicting the software effort compared to the other four models.

Table 4. The equations considered for comparison with ANN model.

Model Name	Equations
Walston - Felix model	$\mathrm{Effort} = 5.2(\mathrm{KLOC})^{0.91}$
Halstead model	$\mathrm{Effort} = 0.7(\mathrm{KLOC})^{1.5}$
Doty model	$\mathrm{Effort} = 5.288\ (\mathrm{KLOC})^{1.047}$
Bailey-Basili model	$\mathrm{Effort} = 5.5 + 0.73(\mathrm{KLOC})^{1.16}$

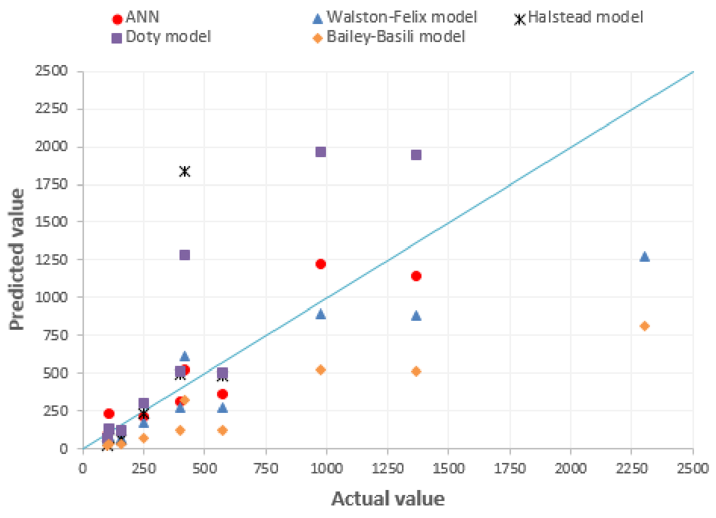

Figure 5. Predicted versus actual effort estimation for testing data set.

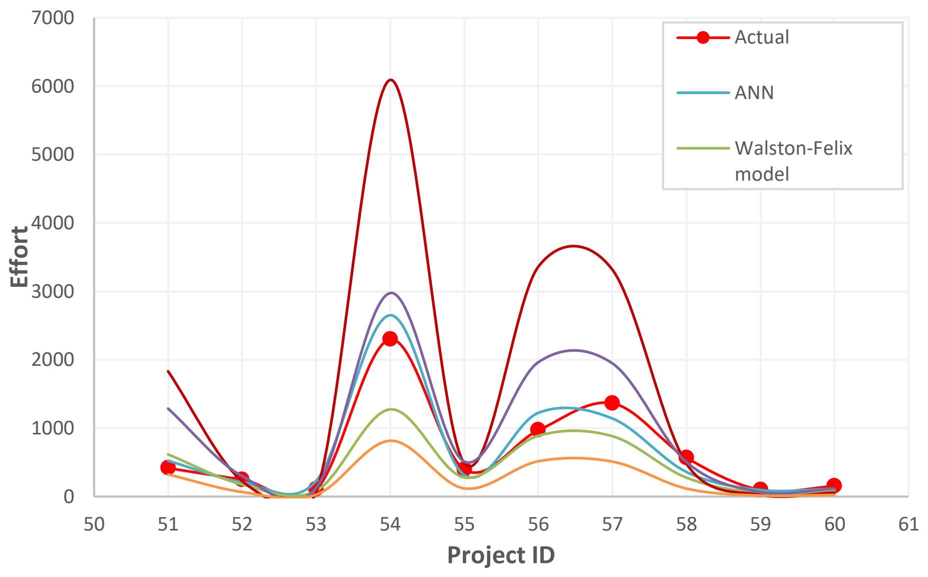

Figure 6. Comparison of the performance of the predicted and actual effort values for test period.

Table 5. Summary results.

Performance Criteria	**Model Used**				
	ANN	**Walston-Felix model**	**Halstead model**	**Doty model**	**Bailey-Basili model**
RMSE	178.13	380.44	1609.38	503.72	590.30
MMRE	0.299	0.373	1.108	0.514	0.650

4. CONCLUSION

This study investigated using artificial neural networks (ANN) for estimating the software development effort. Three variables representing the CPLX, PCAP, and KLOC are considered as input parameters to predict the output (effort). The NASA dataset is applied for building the predictive model. For the training and testing phases of the model, 50 and 10 projects are used, respectively. The optimum ANN model is chosen after experimenting different models structure. The best ANN model was found to be ANN (3-10-9-1). The evaluation is conducted using three error indexes, such as the MMRE, RMSE, and the correlation coefficient R. The amount of MMRE, RMSE, and R for the test period are 0.299, 178.13, and 0.976, respectively. The results of the ANN method demonstrate a satisfactory agreement with the actual data. Furthermore, the proposed ANN model showed the most optimized results compared with the other existing models such as the Walston-Felix model, Halstead model, Doty model, and Bailey-Basili model. Finally, it could be declared that the ANN can be used as a reliable, and efficient approach in the modeling and prediction of software development effort. For Future work, new methods and models can be done to estimate software effort based on machine learning algorithms.

REFERENCE

Aggarwal, K., Singh, Y., Chandra, P., & Puri, M. (2005). Evaluation of various training algorithms in a neural network model for software engineering applications. *ACM SIGSOFT Software Engineering Notes, 30*(4), 1–4.

Bhatnagar, R., Bhattacharjee, V., & Ghose, M. K. (2010). Software development effort estimation–neural network vs. regression modeling approach. *International Journal of Engineering Science and Technology, 2*(7), 2950–2956.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ: Prentice Hall.

Jeon, J., & Rahman, M. S. (2008). Fuzzy neural network models for geotechnical problems.

Kaur, J., Singh, S., Kahlon, K. S., & Bassi, P. (2010). Neural network—a novel technique for software effort estimation. *International Journal of Computer Theory and Engineering, 2*(1), 17.

Kumar, K. V., Ravi, V., Carr, M., & Kiran, N. R. (2008). Software development cost estimation using wavelet neural networks. *Journal of Systems and Software, 81*(11), 1853–1867.

Mair, C., et al. (2000). An investigation of machine learning-based prediction systems. *Journal of Systems and Software, 53*(1), 23–29.

MathWorks. (2009). *Neural network toolbox user's guide: For use with MATLAB*.

Mohsin, Z. R. (2021). Application of artificial neural networks in prediction of software development effort. *Turkish Journal of Computer and Mathematics Education (TURCOMAT, 12*(14), 4186–4202.

Mohsin, Z. R. (2021). Comparative study for software effort estimation by soft computing models. *Journal of Education for Pure Science-University of Thi-Qar, 11*(2), 108–120.

Mohsin, Z. R. (2021). Investigating the use of an adaptive neuro-fuzzy inference system in software development effort estimation. *Iraqi Journal for Computer Science and Mathematics, 2*(2), 18–24.

Nassif, A. B., Azzeh, M., Capretz, L. F., & Ho, D. (2016). Neural network models for software development effort estimation: A comparative study. *Neural Computing and Applications, 27*(8), 2369–2381.

Rafiq, M., Bugmann, G., & Easterbrook, D. (2001). Neural network design for engineering applications. *Computers & Structures, 79*(17), 1541–1552.

Rijwani, P., & Jain, S. (2016). Enhanced software effort estimation using multi-layered feed-forward artificial neural network technique. *Procedia Computer Science, 89*, 307–312.

Singal, P., Kumari, A. C., & Sharma, P. (2020). Estimation of software development effort: A Differential Evolution Approach. *Procedia Computer Science, 167*, 2643–2652.

Tanarslan, H., Secer, M., & Kumanlioglu, A. (2012). An approach for estimating the capacity of RC beams strengthened in shear with FRP reinforcements using artificial neural networks. *Construction and Building Materials, 30*, 556–568.

Wittig, G. E., & Finnie, G. R. (1994). Using artificial neural networks and function points to estimate 4GL software development effort. *Australasian Journal of Information Systems*, *1*(2).