DOI: http://doi.org/10.32792/utq.jceps.11.01.010

Comparative Study for Software Effort Estimation by Soft Computing Models

Zainab Rustum Mohsin

Received 30/08/2021 Accepted 20/10/2021 Published 11/11/2021

Abstract:

Accurate estimation of software development effort has become a crucial for effective projects planning. It is a very challenging task for project teams to predict the development effort required in the initial phases of a software project. Software estimation prior to development process can decrease the risk and enhance the project's success rate. Although numerous traditional and machine learning models have been proposed for software effort estimation over the past decade, the level of accuracy is not satisfactory enough. The objective of this study is to assess the capability of using two different soft computing methods namely artificial neural networks (ANNs) and adaptive neuro-fuzzy inference system (ANFIS) to estimate the software effort. COCOMO dataset is used to test and train the proposed models. Mean magnitude of relative-error (MMRE) and correlation coefficient (*R*) were used as assessment criteria. It was concluded that both models can satisfactorily estimate software development efforts, but ANFIS model has outperformed the ANN model in two statistical indicators: MMRE and correlation *R*. It is recommended that ANFIS can be used as a predictive model for software effort estimation.

Keywords: Software effort estimation, Soft Computing, ANFIS, ANN, COCOMO.

Software effort estimation has always been a significant challenge for software teams and companies that should be considered in the initial stages of a software project. Accuracy in software effort estimation is critical for software project success and risk reduction. At the same time, each project is of a particular nature that making it much more difficult to estimate the necessary effort for completion and making task predictions more challenging. Effort estimation is a process for predicting the development time and cost needed to develop a software process or product. Proper estimating may involve accurately predicting software costs in budgeting, planning, controlling, and eventually managing the project efficiently. Furthermore, estimating effort can be useful in the determination of the resources required in the future (McConnell, 1996). During software design and development, estimating cost and time accurately is essential for the allocation of resources and reasonable planning. Project success and failure depend on project planning because in this phase the time and budget constraints required to complete the project successfully are estimated (Idri & Abnane, 2017).

Underestimating the effort can cause poor quality of software which eventually may result in project failure, while overestimating the effort can prevent resources which could have been allocated elsewhere in a timelier manner. Accordingly, the success of any software project seems to be highly dependent on the accuracy of its effort estimation. The concept of software effort estimation began to gain attention with the emergence of the computer industry in 1940, and study in this field is still ongoing (Zaid, Selamat, Ghani, Atan, & Wei, 2008).

Different techniques are used in software cost estimation and these techniques can be generally classified into two categories, Algorithmic and Non-Algorithmic. Algorithmic techniques are based on mathematical equations to predict software cost. Examples of algorithmic approaches include Constructive Cost Model (COCOMO) model (Boehm, 1981), Software Life Cycle Management (SLIM) model (Putnam, 1978) and Function Points Analysis (Albrecht & Gaffney, 1983). The non-algorithmic methods are based on new techniques such as regression models, Halstead model, expert judgment, and all machine learning approaches etc., (Edinson & Muthuraj, 2018).

Recently, data-driven models, such as adaptive neuro-fuzzy inference systems (ANFIS) and artificial neural network (ANN) have gained a significant attention from researchers to model the complex relationship between effort and software attributes. Soft Computing (SC) comprises of a variety of complementary techniques like ANN, fuzzy system (FS) and ANFIS. The benefits of adopting the SC methods over other techniques arise from their ability to self-learn from the data and thereby minimize error.

Rijwani and Jain, (Rijwani & Jain, 2016) implemented ANN approach to predict software development efforts. The COCOMO dataset was used for training and testing the proposed model. Their model indicated good performance for the considered prediction. Nassif et al., (Nassif, Azzeh, Capretz, & Ho, 2016) investigated four different types of ANN technique for software development effort estimation. They used the multi-layer perceptron, general regression neural network, radial basis function neural networks, and cascade correlation neural networks and finally found out that the cascade correlation neural network had better performance than the other three methods. Kamal *et al.*, (Kamal & Nasir, 2013) modelled the software effort estimation with the help of fuzzy logic approach. Triangular membership function was used to express linguistic fuzzy values in the COCOMO II model. They obtained good results by comparing the proposed model with the COCOMO II and Alaa Sheta Model. VinayKumar et al., (Kumar, Ravi, Carr, & Kiran, 2008) used wavelet neural network (WNN) method for the software development effort estimation. Mohsin, (Mohsin, 2021a) investigated the software development effort estimation using ANN approach. The proposed model confirmed that ANN could be used for prediction purposes. Idri et al., (Idri & Abnane, 2017) presented a model to estimate the software development effort using Fuzzy Analogy on the COCOMO’81 datasets. Their results indicated that Fuzzy Analogy was a promising approach for software development effort estimation. Wittig and Finnie, (Wittig & Finnic, 1994) applied artificial neural network approach for software development effort estimation. Mohsin, (Mohsin, 2021b) used ANFIS to predict the software effort estimation and found that ANFIS was a suitable approach to prediction. Reddy and Raju, (Reddy & Raju, 2009) investigated the ability of the fuzzy system to model the effort estimation required for software development. The proposed model was based on COCOMO dataset. Karimi and Gandomani, (Karimi & Gandomani, 2021) used a hybrid of ANFIS and differential evolution algorithm (ANFIS-DE) to predict the software development effort. They obtained better performance by comparing the ANFIS-DE model with the other models. Nanda *et al.*, (Nanda & Soewito, 2016) Proposed a model for the software effort estimation using hybrid of particle swarm optimization and ANFIS approaches (PSO- ANFIS). They used NASA datasets to develop the proposed model. The parameters cost driver contain of 17 attributes which were optimized utilizing PSO. Moosavi et al., (Moosavi & Bardsiri, 2017) used a hybrid of ANFIS and satin bower bird optimization algorithm (SBO) methods for modelling the software effort estimations. The hybrid ANFIS-SBO model is an optimized neuro-fuzzy-based estimation model which can produce accurate estimations in software project. The proposed model was able to get better performance than other models in all of the evaluation metrics. Praynlin and Latha (Edinson & Muthuraj, 2018) employed ANFIS and Elman NN to predict the software development effort. Their results demonstrated that the ANFIS model generated a higher degree of accuracy than the Elman NN model.

Finally, in the last decades, soft computing approaches particularly ANN and ANFIS were the prominent techniques for developing predictive models. Therefore, newer soft computing-based techniques can always be used to develop more accurate predictive precision models.

In this vision, this paper presents a case study using artificial neural networks (ANNs), and adaptive neuro-fuzzy inference system (ANFIS) model the effort estimation required for software development. The prediction results obtained from both methods were compared. The COCOMO data set (Boehm, 1981) was used for constructing the proposed models.

2. Prediction Techniques

2.1. Artificial Neural Network (ANN)

An artificial neuron is a computational tool which simulated the natural neurons. It is dependent on a database that contains both inputs and outputs. Figure 1 shows the basic structure of the neural network consisting of input data, weights, transfer functions, activation functions, threshold and output.

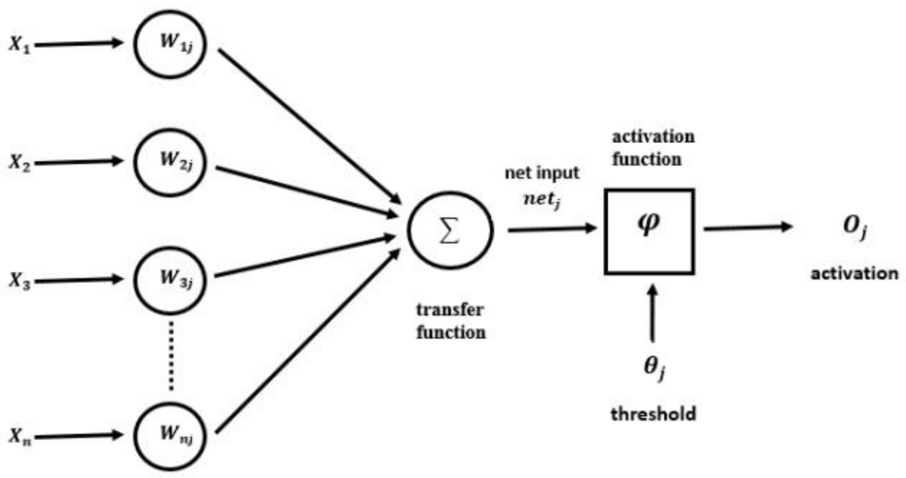

Fig.1. Typical structure of neural network (Haykin, 1999).

The artificial neurons are fed by numbers of inputs. Each node has a computational process that involves multiplying each input value by its corresponding connecting weights, summing up their product, and then the non-linear transfer function is used to determine the desired output (Jeon & Rahman, 2008), as presented in Equation (1).

Total activation: $\text{net}\, j = \sum_{i=1}^{n}(w_{ij})x_i\text{-}T_j$ (1)

Where *j* is the actual neuron number, *n* is the number of nodes, w_{ij} is the weight of the node, x_i is the input variable , and T_j is the bias of the nodes.

The feed forward backpropagation algorithm is carried out in the considered ANN simulations. An ANN architecture consisting of three layers: an input layer which presents network input variables, one or more hidden layers, and an output layer consists of a single node that gives the software effective estimation. The trials showed that the network with two-hidden layer outperforms the network with single-hidden layer. The optimum number of nodes in the hidden layers was determined by using a trial and error method.

The training process in the backpropagation neural network involves feeding a set of input - output datasets. The main goal of the training step is to adjust the connection weights to a reasonable level by minimize the errors between the target output and the ANN output. The number of hidden layers, hidden neuron numbers, transfer functions, and normalization of data are also selected in order to obtain the best model performance. After the minimization of errors, testing is conducted using a separate set of data that not used during the training step to evaluate the efficacy of the predictive model. This process is known as generalization of the network. During this phase there is no additional weight adjustment.

2.2. Adaptive Neuro-Fuzzy Inference System (ANFIS):

The adaptive neuro-fuzzy inference system, first proposed by Jang (Jang, 1993) , is a hybrid system of neural network and fuzzy logic systems. Neural network is capable of accurately learning virtually the nonlinear function through a learning process whereas fuzzy logic allows for a better representation of a given system behavior through a set of fuzzy IF-THEN rules (Mohandes, Rehman, & Rahman, 2011). ANFIS is a new improved tool and data-driven modeling technique aims to determine the behavior of imprecisely dynamical complex systems (Kim & Kasabov, 1999).

For simplicity, we assume the fuzzy inference system has two inputs (X_1& X_2), and one output (*y*). A typical rule set with two fuzzy if-then rules for a first-order Sugeno fuzzy model can be explained as (Takagi & Sugeno, 1985):

Rule 1: If X_1 is A_1 and X_2 is B_1. then $y_1 = f_1 = m_1X_1 + n_1X_2 + q_1$

Rule 2: If X_1 is A_2 and X_2 is B_2. then $y_2 = f_2 = m_2X_1 + n_2X_2 + q_2$.

Where: A_1 and A_2 are membership functions for input X_1, B_1 and B_2 are membership functions for input X_2, and m_1, n_1, q_1 , m_2, n_2, q_2 are the design parameters for the output function that are determined during the training process of the ANFIS model.

Figure 2 presents the ANFIS architecture. The characteristics of each layer in the architecture are described as follows:

Layer 1: Input nodes, the main function of this layer is fuzzification of the received inputs using membership functions. The output of the *i*th node is defined by:

$$O_i^1 = \mu_{A_i}(x) \qquad \text{for } i = 1,2 \qquad (2)$$

where *x* = input to the *i*th node, A_i = linguistic label associated with the node, and μ_{A_i} = membership function. The parameters in this layer are generally defined as premise parameters.

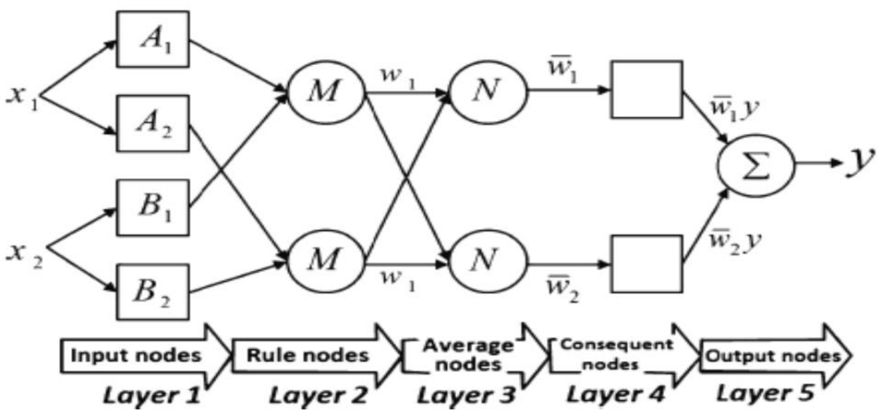

Fig. 2. The architecture of an ANFIS approach (Jang, 1993).

Layer 2: Rule nodes, the nodes in this layer are fixed node and denoted as M (Figure 2). In each node the incoming signals are multiplied, and the output O_i^2 that represents the firing strength of a rule is calculated as follows:

$$O_i^2 = w_i = \mu_{Ai}(x)\,\mu_{Bi}(x). \qquad \text{for } i = 1,2 \qquad (3)$$

Layer 3: Average nodes. All the nodes in this layer are fixed nodes labeled by N (Figure 2). The *i*th node in this layer computes the ratio of the firing strength of the *i*th rule to the sum of firing strengths of all the rules as follows:

$$O_i^3 = \overline{w}_i = \frac{w_i}{w_1 + w_2}. \qquad \text{for } i = 1,2 \qquad (4)$$

Layer 4: Consequent nodes. All the nodes in this layer are adaptive node with a node function

$$O_i^4 = \overline{w}_i\, y_i = \overline{w}_i\,(m_i X_1 + n_i X_2 + q_i), \qquad \text{for } i = 1,2 \qquad (5)$$

where $\overline{w}_i$ = output of layer 3, and $\{m_i \,.\, n_i .\, q_i\} =$ consequent parameter set .

Layer 5: Output nodes. The single node in this layer is a fixed node denoted by Σ, with node function to compute the overall output of the ANFIS:

$$O_i^5 = \sum_i \overline{w}_i\, y_i = \frac{\sum_i \overline{w}_i\, y_i}{\sum_i \overline{w}_i} \qquad \text{for } i = 1,2 \qquad (6)$$

More details about the ANFIS algorithm can be found in (Jang, 1993).

3. Data Description

In this study, the COCOMO 81 dataset (Boehm, 1981) was used in order to develop the ANN and ANFIS models. COCOMO81 was chosen because it is a public domain database, which has already been used for different methods. These data are consisting of 63 software projects. Each project has 17 features: the software size is defined in KDSI (Kilo Delivered Source Instructions) and the remaining 15 features are defined by a 6 linguistic values: 'very low', 'low', 'nominal', 'high', 'very high' and 'extra high'. Table 1 displays the cost driver variables considered in this study. These 16 features are associated to the software development environment like the people's experience that involved in the software project, the method of development and the time and storage restrictions that imposed on the software. The output of the ANFIS and ANN models is the software effort, which is measured in man-months. The data was randomly divided into two sets: 52 (83%) were used for models training, and the remaining 11(17%) were used for testing the proposed models.

Table 1: COCOMO81 cost drivers

Variable	Full name
ACAP	Analyst capability
TOOL	Use of software tools
AEXP	Applications experience
RELY	Reliability
MODP	Use programming modern practices
DATA	Database size
VEXP	Experience with virtual machine
PCAP	Programmers capability
CPLX	Process complexity
LEXP	Programming language experience
TIME	Restriction of time
TURN	Computer turnaround
SCED	Schedule constraint
VIRT	Virtual machine volatility
STOR	Main storage constraint
RVOL	Requirements volatility
ADJKDSI	Software size

4.Performance Measures:

The predictive accuracy of the developed models was evaluated using the following performance indicators:

a. The mean magnitude of relative error (MMRE): MMRE, which is the mean measurement of the absolute values of the relative errors over an entire data set (Baker, 2007).

$$\text{MMRE} = \frac{1}{N}\sum_{1}^{N}\frac{|\text{Actual Effort-Predicted Effort}|}{\text{Actual Effort}} \qquad (7)$$

where N is the number of estimates.

b. The Correlation coefficient (*R*): *It* is used to measure the strength of the linear relationship between the actual and estimated values.

$$R = 1\text{-}\sqrt{\frac{\sum_{i=1}^{N}(\text{Actual Effort-Predicted Effort})^2}{\sum_{i=1}^{N}(\text{Actual Effort})^2}} \qquad (8)$$

The *R* values range between -1 and +1, with the best models that give correlation *R* values that are as close to +1 as possible.

5. Results and Discussion:

Modeling using ANFIS and ANN approaches, contains of four stages: (a) data preprocessing, (b) building the model (structure), (c) training, and (d) testing of ANFIS and ANN models. MATLAB software program has been utilized for implementing the proposed models. There are 16 inputs parameters and effort as output variable that was utilized for building the predictive models, which displayed in Table1.

5.1 Generation of the ANN model:

The ANN model was developed utilizing neural network toolbox ("Neural network toolbox user's guide: for Use with MATLAB," 2009) of MATLAB software. For training neural network architecture, the feed forward backpropagation algorithm was used. The activation function for the input and hidden layers was logistic sigmoidal function, while the output layer used a linear function. To begin training the network, the number of hidden layers, number of neurons in hidden layer, and number of epochs should be specified.

For the best developed ANN model, mean square error (MSE) have been used to evaluate the prediction performance of the proposed model during training and testing phase. After trial and error process, the network with two hidden layer showed better performance than the network with a single hidden layer. The optimum number of neurons in the hidden layers giving the optimum ANN framework with the minimum MSE was determined as nine neurons in the first hidden layer and twelve neurons in the second hidden layer.

The network was continually trained by update weights until it reached a final error of 0.026 after 47 epochs. The predictions results obtained from the ANN model using the test dataset are given in Table 2.

Table 2. Estimated effort using ANN technique.

Project ID	Actual effort	Computed effort with ANN
6	43	8.32
12	201	269.15
21	2455	1023.29
24	453	1380.38
30	5.9	6.31
36	55	16.22
43	83	97.72
48	1272	478.63
54	20	23.99
61	50	26.92
62	38	33.88

Figure 3 display a comparison of the actual and predicted effort from ANN model during the testing phase. Further Table 3 tabulates the correlation *R* of the ANN model for training and testing datasets. The values of *R* were 0.94 for training and 0.909 for testing phase.

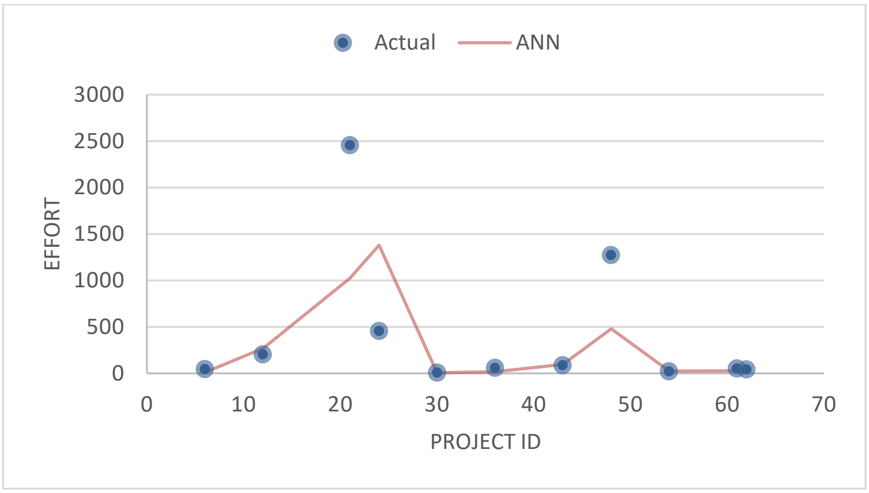

Fig.3. Comparative charts of actual and estimated effort values using ANN model for testing datasets.

Table 3. correlation coefficient *R* for ANN model.

Model	*R*	
	Training	Testing
ANN	0.94	0.909

5.2 Generation of the ANFIS Model:

ANFIS models can be composed of various structures that govern the output and estimation performances. The subtractive clustering method introduced by Chiu (Chiu, 1994) was used to produce the FIS structure using the MATLAB function (*genfis2*). The *genfis2* generates a fuzzy inference system (FIS) using subtractive clustering method and requires separate pairs of input-output data as input argument. The subtractive clustering procedure is as follows: first, choose data point of the first cluster that has the highest potential. The next cluster and its center are obtained by eliminating all data points inside the radial distance of the first cluster. This process is continuing till all the data points within the radii of a cluster center are clustered (Chiu, 1996). The range of influence of a cluster center in each of the data dimensions called the cluster radius (r). The cluster radius is an essential parameter for calculating the number of clusters and it is determining by using the trial and error method. A large value of radius may result in a coarser model with less number of clusters, while A smaller value of radius may cause a large number of clusters with a finer model.

The commonly used fuzzy inference systems are Mamdani and Sugeno (Mamdani & Assilian, 1975; Sugeno, 1985). In this paper, the Sugeno-method fuzzy inference system was used. The type and the number of membership functions were investigated when the prediction results of the training and testing process reach a satisfactory level depending on the MSE.

The ANFIS model have been constructed using the subtractive clustering method and trained with 52 datasets from the COCOMO dataset. Then, the proposed model has been tested to evaluate the efficiency of the trained ANFIS model; 11 datasets have been utilized for this purpose. Various ANFIS models with a different values of cluster radius and with a different number of epochs have been investigated. After extermination various learning algorithms with a different number of epochs, the optimum correlations were archived using a hybrid learning algorithm (a combination of back-propagation algorithm and least squares method for learning the premise and consequent parameters respectively). Membership function of type Gaussian membership functions (*gaussmf*) were used for input variables and linear function for output parameters.

However, based on the previously discussed criteria, it is observed that the of model with r = 0.55 and number of epochs = 300 has the best performance compared to the others and it is selected as the final ANFIS model. The proposed model achieved final errors of 221.74 for the training stage and 126.08 for the testing stage after 300 epochs. Table 4 presented the development effort estimates for each testing dataset. Figure 4 display a comparison of the actual and predicted effort from ANFIS model during the testing stage. Table 5 presents the correlation *R* of the ANFIS model for training and testing datasets. The predictions tend to be extremely good with correlation coefficient *R*. The values of *R* were 1.0 for training and 0.99 for testing phase.

Table 4. Estimated effort using ANFIS technique.

Project ID	Actual effort	Computed effort with ANFIS
6	43	59.9
12	201	76.44
21	2455	2449.13
24	453	171.83
30	5.9	7.14
36	55	15.7
43	83	90.26
48	1272	992.54
54	20	14.71
61	50	48.84
62	38	22.47

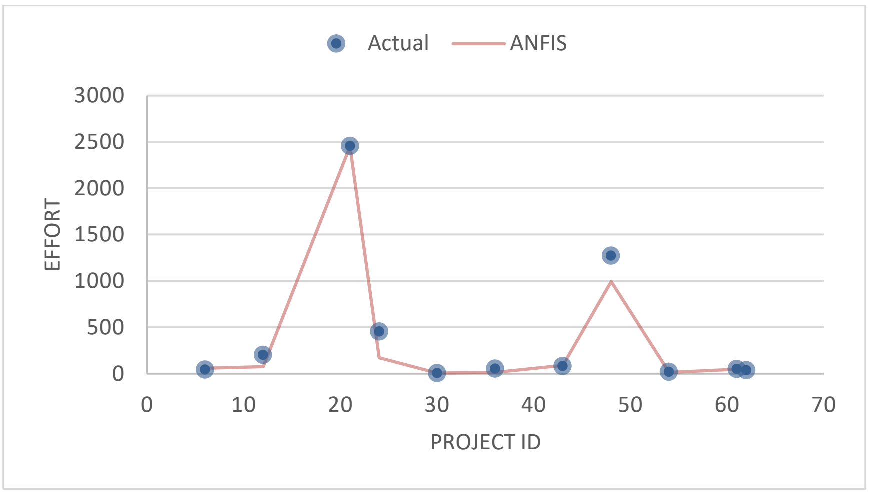

Fig.4. Comparative charts of actual and estimated effort values using ANFIS model for testing datasets.

Table 5. correlation coefficient *R* for ANN and ANFIS models.

Model	*R*	
	Training	Testing
ANFIS	1.0	0.99

5.3 Comparison of the techniques

A head-to-head comparison of the prediction accuracy of the ANN and ANFIS models for testing dataset are shown in Figure 5. It can be observed from this figure that the prediction values of the ANFIS model match well with the actual datasets and it outperformed the ANN model.

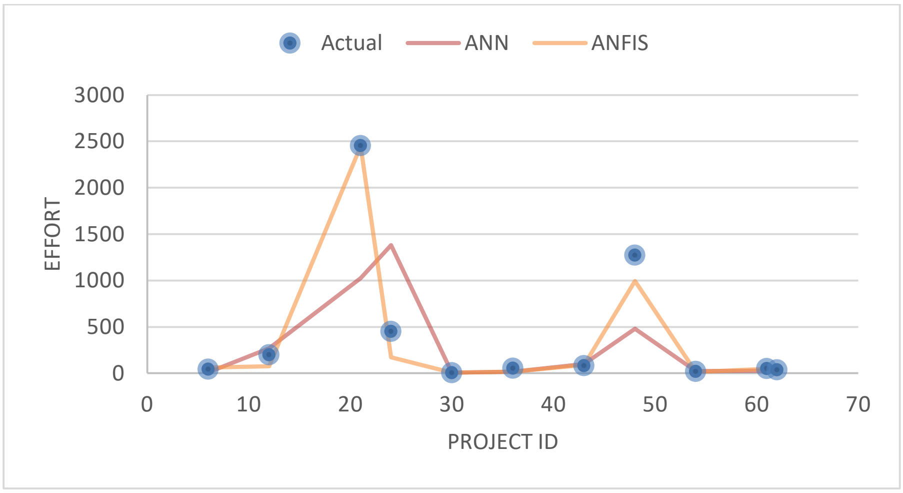

Fig.5. A comparison of performance for ANN and ANFIS models.

Table 6 summarizes the MMRE and correlation *R* values resulting from the ANN and the ANFIS models, and results obtained by Kemerer (Kemerer, 1987)with Function Points, COCOMO-Basic, and SLIM models. According to Table 4, although ANFIS outperforms ANN, both ANN and ANFIS have produced a scientifically higher MMRE and correlation *R* values than the other models.

Table 6. Comparison of performance measured for proposed models with traditional models.

Model	MMRE	*R*
ANN	55.65	0.908
ANFIS	32.4	0.989
COCOMO Basic	610	0.70
Function Points Analysis	103	0.58
SLIM	772	0.89

In terms of correlation *R*, the ANFIS model (0.989) outperforms the COCOMO (0.70), the SLIM and the Function Point Analysis models (0.89 and 0.58, respectively), in Kemerer's experiments. On the MMRE dimension, the ANFIS significantly better than the other methods, and outperforms the ANN model. The reason behind superior positive results of ANFIS may be attributed to its structure and the ability to eliminate noisy data, ANFIS model utilizes "IF–THEN" rules to generate an output for each rule, allowing it for learning from the data (Tofigh, Rahimipour, Shabani, & Davami, 2015). The results revealed that the ANN and ANFIS models were able to be effective in software effort estimation.

6. Conclusion:

This study evaluated two different approaches of soft computing namely ANN and ANFIS to estimate the effort of software projects. The Boehm's COCOMO dataset have been utilized for training and testing the predictive models. The performance metrics used were MMRE and correlation coefficient R. It was concluded that the ANN and ANFIS models provided significantly better effort estimations than the Function Points, COCOMO-Basic, and SLIM methods. However, ANFIS model outperforms the ANN model on all evaluation criteria. In other words, the ANFIS model resulted in a better match with the actual values. This can be attributed to the architecture of ANFIS. The ANFIS captures the benefits of the learning capability of neural network and the simplifying function of fuzzy reasoning, resulting in a high ability to eliminate noise (Rajaee, Mirbagheri, Zounemat-Kermani, & Nourani, 2009). ANFIS is recommended for utilize as reliable and simple tools for the software effort estimation.

References:

[1] Albrecht, A. J., & Gaffney, J. E. (1983). Software function, source lines of code, and development effort prediction: a software science validation. *IEEE Transactions on software engineering*, *9*(6), 639-648.
[2] Baker, D. R. (2007). *A hybrid approach to expert and model based effort estimation*: Citeseer.
[3] Boehm, B. (1981). *Software engineering economics*: Prentice-Hall, Englewood Cliffs, NJ.
[4] Chiu, S. L. (1994). Fuzzy model identification based on cluster estimation. *Journal of Intelligent & Fuzzy Systems*, *2*(3), 267-278.
[5] Chiu, S. L. (1996). Selecting input variables for fuzzy models. *Journal of Intelligent & Fuzzy Systems*, *4*(4), 243-256.

[6] Edinson, P., & Muthuraj, L. (2018). Performance analysis of FCM based ANFIS and ELMAN neural network in software effort estimation. *Int. Arab J. Inf. Technol., 15*(1), 94-102.

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

[8] Idri, A., & Abnane, I. (2017). Fuzzy analogy based effort estimation: An empirical comparative study. *2017 IEEE International Conference on Computer and Information Technology (CIT)*, 114-121.

[9] Jang, J.-S. (1993). ANFIS: adaptive-network-based fuzzy inference system. *IEEE transactions on systems, man, and cybernetics, 23*(3), 665-685.

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

[11] Kamal, S., & Nasir, J. A. (2013). A fuzzy logic based software cost estimation model. *International Journal of Software Engineering and Its Applications, 7*(2), 7-18.

[12] Karimi, A., & Gandomani, T. J. (2021). Software development effort estimation modeling using a combination of fuzzy-neural network and differential evolution algorithm. *International Journal of Electrical & Computer Engineering (2088-8708), 11*(1).

[13] Kemerer, C. F. (1987). An empirical validation of software cost estimation models. *Communications of the ACM, 30*(5), 416-429.

[14] Kim, J., & Kasabov, N. (1999). HyFIS: adaptive neuro-fuzzy inference systems and their application to nonlinear dynamical systems. *Neural networks, 12*(9), 1301-1319.

[15] 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.

[16] Mamdani, E. H., & Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller. *International journal of man-machine studies, 7*(1), 1-13.

[17] McConnell, S. (1996). *Rapid development: taming wild software schedules*: Pearson Education.

[18] Mohandes, M., Rehman, S., & Rahman, S. (2011). Estimation of wind speed profile using adaptive neuro-fuzzy inference system (ANFIS). *Applied Energy, 88*(11), 4024-4032.

[19] Mohsin, Z. R. (2021a). APPLICATION OF ARTIFICIAL NEURAL NETWORKS IN PREDICTION OF SOFTWARE DEVELOPMENT EFFORT. *Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12*(14), 4186-4202.

[20] Mohsin, Z. R. (2021b). 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.

[21] Moosavi, S. H. S., & Bardsiri, V. K. (2017). Satin bowerbird optimizer: A new optimization algorithm to optimize ANFIS for software development effort estimation. *Engineering Applications of Artificial Intelligence, 60*, 1-15.

[22] Nanda, S., & Soewito, B. (2016). Modeling software effort estimation using hybrid PSO-ANFIS. *2016 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, 219-224.

[23] 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.

[24]Neural network toolbox user's guide: for Use with MATLAB. (2009).

[25] Putnam, L. H. (1978). A general empirical solution to the macro software sizing and estimating problem. *IEEE Transactions on software engineering, 4*(4), 345-361.

[26] Rajaee, T., Mirbagheri, S. A., Zounemat-Kermani, M., & Nourani, V. (2009). Daily suspended sediment concentration simulation using ANN and neuro-fuzzy models. *Science of the total environment, 407*(17), 4916-4927.

[27] Reddy, C. S., & Raju, K. (2009). An improved fuzzy approach for COCOMO's effort estimation using gaussian membership function. *Journal of software, 4*(5), 452-459.

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

[29] Sugeno, M. (1985). An introductory survey of fuzzy control. *Information sciences, 36*(1-2), 59-83.

[30] Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE transactions on systems, man, and cybernetics*, *15*(1), 116-132.

[31]Tofigh, A. A., Rahimipour, M. R., Shabani, M. O., & Davami, P. (2015). Application of the combined neuro-computing, fuzzy logic and swarm intelligence for optimization of compocast nanocomposites. *Journal of Composite Materials*, *49*(13), 1653-1663.

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

[33] Zaid, A., Selamat, M. H., Ghani, A., Atan, R., & Wei, K. (2008). Issues in software cost estimation. *IJCSNS Int J of Computer Science and Network Security*, *8*(11), 350-356.