Recurrent Neural Network based Prediction of Software Effort

Lujain A. Hussein
University of Basra, Iraq

Kulood A. Nassar
University of Basra, Iraq

Maysaa A. Naser
University of Basra, Iraq

ABSTRACT
The enormous efforts of software systems and unexpected efforts in the late phases of software development in software engineering field led to using methods to estimate software effort at early stages of software preparing phases. Therefore, the question remains how can develop an estimation method to be more accurate and gives a prediction for future software efforts. This paper presents a proposed method for software effort prediction, to enhance software effort estimation phase. The proposed method utilizes feed-forward neural network in recurrent fashion to make a prediction and adapt to handle with varying software types in software engineering. The proposed method (RFFNN) used to enhance the results of ordinary software effort estimation methods, RFFNN gives more efficient results by making a prediction for future software efforts.

General Terms
Software engineering

Keywords
Software efforts estimation, Soft Computing, Recurrent Feed-Forward Neural Network, Neural Networks, Software effort prediction.

1. INTRODUCTION
A discipline that called Software engineering (SE), its aim is solving problems as business problems by designing software systems and developing it. The world cannot be run without using the software. Systems that depend on computers are controlling National infrastructures and most electrical and other products. The computerizing of manufacturing industry is needed nowadays. Entertainment like computer games, music, and television is software intense. Therefore, the value of software engineering arises for both the national and international societies. However, physical constraints lack makes software systems to be complex, expensive to develop, and difficult to be understandable, so that software systems have different types, from simple, complex, and worldwide systems. Measurement and software estimation are important processes in critical software engineering [1], [2].

Cost estimation considered as an important cost calculation procedure. It must be complete at the first stages of any project to determine the involving of functions across the processes and to avoid exceeding the limits of the available budget [3], [4]. Projects could be software projects, construction (building) projects, and many other types. The calculating process of effort that is needed to build the software and determine the complexity of it is known as effort estimation. Incorrect, both underestimates or overestimates of the effort required, complicate scheduling, and management result in over budgets and late in the projects. These reasons represent why effort estimation is somewhat known as sort of critical part [4], [5].

1.1 Software effort
The effort of software is a forecasting critical process used to give a real effort amount that is required for software projects to reach project's completion time. Producing efficient software project is demanding and its activities are essential for developing software. It holds good when the process of development is started to meet today's demands of industry. Now a software that has lesser time with low cost and good quality is what people want [11].

1.2 Categories of Effort Estimation
The methods of effort estimation classified into these three primary categories such as, Analogy based, Algorithmic method, and Expert judgment [7].

Algorithmic modeling developed by using information of historical effort which relates to some metrics like size. Estimation of desired effort is done by that metric. This model involves using mathematical models as COCOMO.

Expert judgment the expert's experience here is important and takes the first priority. The accuracy of expert-based estimation model is low.

Analogy Estimation this technique is practicable when there are completed projects found in that application domain. Estimation of the new project cost is done by using an analogy with same domain completed projects.

Neural Networks NNs are defined as computational tools for modeling that have an acceptance in several disciplines the aim is modeling real-world hard problems. ANNs have widely used for several tasks, like nonlinear control, pattern classification, function approximation, time series predictions, and efforts or cost estimation. The idea behind ANNs was conceived for modeling bio-physiology in the human brain to understand how it works. The aim was to create models that have the ability to emulate the process of human reasoning. Many patches of starting works in NN is done by physiologists, not engineers [8], [9]. Feed-forward Neural Networks are the simplest and first type of ANNs. In FFNNs, as shown in Figure (1), the data always moves in just one direction, forward from the input nodes, throughout the hidden nodes then to the output nodes and no loops in that networks. These networks utilized many neurons (nodes) and contain no any feedback paths in them. They are the widely used NNs, particularly in systems and controls. Multilayer NNs have input layer, hidden layers, and output layer (no interconnections between the nodes in same layers), they called a hidden and exist in between two layers, input and also output layer [10], [11].

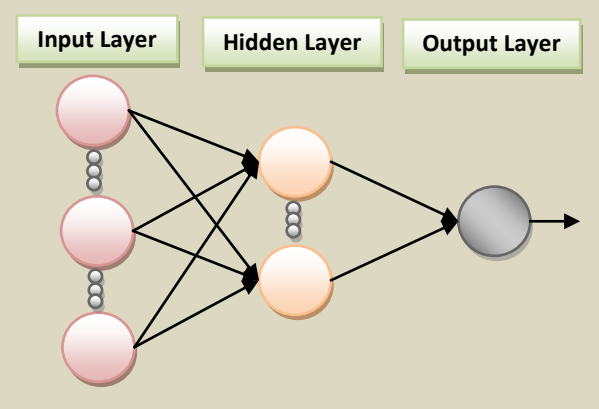

Fig 1: Feed-Forward Neural Network

2. LITERATURE REVIEW

This section highlights on the past research in the area of software effort estimation.

J. Maximino and R. Lyne [12], Develop an Artificial Neural Networks (ANN) model with backpropagation structure for cost estimation of Philippine building projects. They collected 30 building projects data then divided it randomly to three sets: 20% used for validating performance, 60% used for training, and 20% as a network generalization independent test. Six inputs parameters were used that are a number of floors, floor area, the number of basements, concrete volume, formworks area, and reinforcing steel weight.

M. Kumar and S. Abbinaya [13], concentrated on two techniques, function point and use case points to use them for estimation. By using decision table, they compare these methods to test them for acquiring accurate results. Neural Networks (NN) is applied for decision table training. NNs have used also for comparing the produced trained efforts with real effort. Function point and use case points inputs are trained by the NN algorithm. Decision table represents a place to store trained inputs.

S. Gotovac and H. Karna [5], Proposed an approach for improving effort estimation by using NNs as Bayesian networks (BN). They test major tree's entities in the work items, the estimation process, and estimators. Analysis depended on data that is collected from many software projects which all executed in the Croatian software corporation. They found BNs is a suitable way in the software estimation modeling.

Abran, A. Idrii, and A. Hasani [7], Evaluated Radial Basis Function Neural Network (RBFN) construction based on fuzzy and hard C-mean clustering algorithm by using the cross-validation approaches. The important objective of the study is investigating whether the RBFN model that learned from training data can estimate efforts of unseen data. they evaluate it using historical datasets which are ISBSG R8, and Constructive Cost Model (COCOMO) COCOMO8.

S. Kumar and P. Agrawal [14], Presented a simple software effort estimating method. They used sufficient minimum parameters which are easy to be identified at the early time of a project. To improve estimation accuracy, they introduced weight factor and calculate it by an expert learning method.

N. Goyal and M. Bisi [15], Proposed a technique with single layer NN (SLP) that predict software efforts from quality metrics of software. They used swarm optimization for training the network and used Principal components analysis (PCA) for minimizing input features dimensionality, finally, for optimizing the NN, genetic algorithms are used. A new architecture of ANN is proposed with additional input layer used for encoding. This layer is involved before hidden layers. This new layer used to scale input values of NN.

3. RESEARCH METHODOLOGY

Software effort estimation is an important field in software engineering, traditional algorithmic methods are not accurate enough for estimation, however, non-algorithmic methods appear more accuracy such as feed-forward backpropagation neural networks. In this paper, recurrent feed forward neural network is used to predict software effort related to software development project activities as shown in Fig.(2).

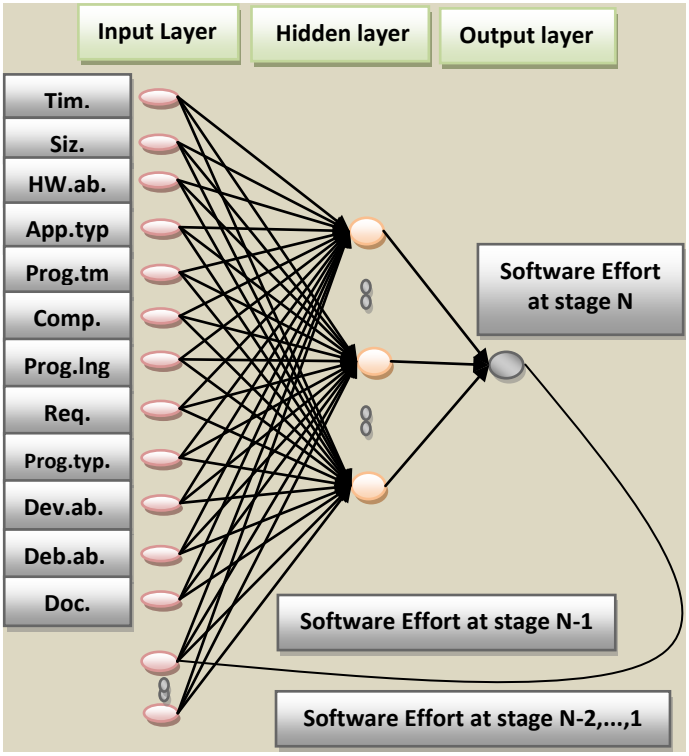

Fig 2: Structure of proposed RFFNN

3.1 Parameters

The inputs of the proposed RFFNN must be selected dependent on which parameters are affecting on estimation, the parameters that were used in this proposed method are 12 parameters as follows.

1- Time (Tim.) - Consists of five scales which are very low, low, medium, high, and very high, it relates to the execution time of past software projects.

2- Size (siz.) - This parameter used to compute software size, it has five scales such as very low, low, medium, high, and very high.

3- Hardware ability (HW.ab.) – This parameter deals with the hardware aspects that relate the software it scales such that very old, old, semi-old, modern, and very modern.

4- Application type (App.typ.)– Contains any types of these scales Platform and management, education, and reference, home and entertainment, operations and professional.

5- Programmers team (Prog.tm.) – This parameter depends on the total number of programmers the software and scales such that very low, low, medium, high, and very high.

6- Complexity – this parameter depends on this equation.

$$(\boldsymbol{Ex.Time} \times \boldsymbol{Num.Line})\ \% \qquad \textbf{(1)}$$

Where, ExTime is the execution time and Num.line is the number of effective lines in the software code and scales such that very complex, complex, medium, simple, and very simple.

7- Programming language (Prog.lng.) – This parameter computed dependent on whether the programming language that used is old such as basic or C, modern such as C++ and MATLAB, or Visual languages such as Visual C++ and Visual Basic.

8- Requirements (Req.) – This parameter depends on whether the software meets the requirements that are needed. It divided to available, semi-available, and not available.

9- Programming type (Prog.typ.) – There are many types of programming such as structural programming, object-oriented programming, etc.

10- Development ability (Dev.ab.) – This parameter divided into two scales which are able to develop, and unable to develop, it depends on the programmer's decision.

11- Debugging ability (Deb.ab.) – This parameter depends on staff judgment of programming errors whether it is able to debug or not.

12- Documentation (Doc.) – This parameter decides whether the project is able to write as a document or not.

3.2 Working Mechanisms

The proposed RFFNN worked in a recurrent fashion where its output at stages N-1, N-2, ..., 1 are entered as inputs of its stage N. Software effort estimation of each software is depending on software efforts of five similar software of a previous time. When the network makes the output of software effort to the first similar software of the target software, the output is considered as an additional input to the network with the twelve main inputs. The parameters of the next similar software to the target software will be entered and the output of the first similar software will be the 13'th parameter in the network. The network will give the second output of software effort and the third output of software effort is depending on two previous outputs as inputs and so on. The prediction process is done by taking parameters of similar software of the main software, the network will learn with these extra parameters to make a prediction. Computing the total effort of the parameters as a desired output to the network is done by the weighted average method as shown in the equations below. In this method, the twelve parameters classified into three classes depending on the number of scales as following:-

Class one - Time, size, hardware ability, application type, programmers team, and complexity which have five scales.

Class two - Programming language and requirements which have three scales.

Class three – Development ability, debugging ability, programming type and documentation which have two scales.
For each class, the following four weighted average equations are done to compute the desired output.

$$\boldsymbol{X_i = PS_i * S_i} \qquad \textbf{(1)}$$

$$\boldsymbol{Z_i = X_i / PSUM_i} \qquad \textbf{(2)}$$

$$\boldsymbol{W_i = PER_i * Z_i} \qquad \textbf{(3)}$$

$$\boldsymbol{E = \sum W_i} \qquad \textbf{(4)}$$

Where,

PS Is the number of parameters in each class.
S Is the number of scales that the parameters have in class (i).
PSUM Is The summation of all scales in each class.
W Is the class weight
PER Is the percentage of each class where, class one is 60% (0.6), class two is 15% (0.15), and class three is 25% (0.25).
E Is the computed effort.

3.3 Simulation Results

Algorithms of training and testing the proposed RFFNN are designed using C++ programming language and backpropagation learning algorithm is used. The proposed RFFNN is training on 20 different patterns. Training patterns contain inputs as 12, 13, ..., 16 parameters and desired output is software efforts. In training stage, 0.9 training factor and 0.5 smoothing factor are used which are selected by trial and error for giving best convergence and less number of epochs.
Points could be clarified:

- Stage means a one training cycle.
- The output of training stage 1 which is 0.4 becomes an additional input of training stage 2.
- The output of training stage 2 which is 0.3 with the output of stage 1 which is 0.4 become additional inputs in training stage 3.
- The output of training stage 3 which is 0.4 with the previous outputs of training stage 1 and 2 become additional inputs in training stage 4.
- The output of training stage 4 which is 0.3 with the outputs of training stages 1, 2, and 3 become additional inputs in training stage 5.
- Stage 5 gives the predicted software effort of the required software that the proposed RFFNN is trained for it on 5 similar patterns in recurrent fashion.
- Through training stage, the proposed RFFNN reach to desired output at less error value, and it takes a number of epochs for that. For example, Fig (3) – shows error value versus a number of epochs for RFFNN output in the table (1).
- Testing is applied on proposed RFFNN. For that, it gives successes rate 100% of 20 training patterns and success rate 95% of 15 unseen training patterns.

Table 1: Results of proposed RFFNN

Stage	Inputs													Output
	Recurrent Parameters	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	Software Effort(SE)
1	0	1	3	4	4	3	3	3	3	3	1	2	2	0.4
2	0 0.4	3	2	2	3	1	2	3	2	3	2	2	1	0.3
3	0 0.4 0.3	4	5	5	4	3	3	2	2	1	2	1	2	0.4
4	0 0.4 0.3 0.4	3	4	3	5	2	3	2	1	1	2	2	2	0.3
5	0 0.4 0.3 0.4 0.3	4	3	3	5	2	2	3	1	2	2	2	1	0.3
Where, P1 – P12 is the twelve parameters from Size to Documentation.														

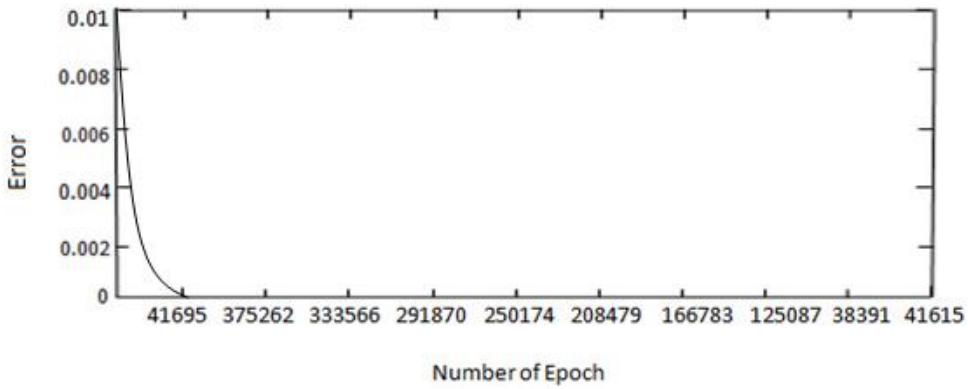

Fig 3: Error versus number of epochs of the proposed RFFNN for software effort in Table 1

4. APPLICATION

In this section, the proposed method is applied to a set of software that used data mining techniques, the affected parameters are taken from this set of software. This application worked on a pack of data like data points or image data where each point is scanned and search for the nearest neighbour for it, for example, the five nearest neighbour points are taken for each point, then a line is drawn between

the point and its nearest neighbours this is done for each data point. It is useful for giving the outlines of the data, for example, if this data is an image data then the outline for the image will appear. This application could be used with image processing and data reduction. The complexity of the application could be increased as it is developed for complex requirements, in this case, it will need a good specification hardware to run on. Figure (4) shows the result of a few number of data points.

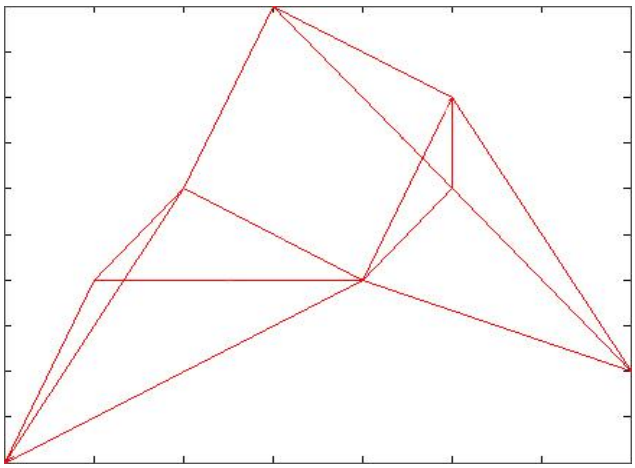

Fig 4: Application results of few data

4.1 Applying The Proposed Method on the Application

The proposed method makes predictions on software efforts by using RFFNN and taking the effective factors of the software systems for using them as neural network parameters as discussed earlier. The twelve parameters are an effective factor for any software to estimate the total effort which ranges from very low to very high. In this section, MATLAB language is used to write the software of application. The application is developed which makes it more complex on time, four software is introduced as following for predicting of the effort of other future software.

Software one – This software takes a set of data points then compute the nearest four neighbors and connect each point with its neighbors as shown in the following code.

Software two – This software is a development where here the data is an image data, convert it to grayscale and make it an array of data points then reduce the number of points and that the most effective points to get image outline which is the black points only any other grayscale points are discarded. The nearest five neighbors are selected as shown in the following code.

Software Three - The software here processes a set of images, convert it to grayscale then takes only the black data points and discards other data, makes the points as an array of data, finally finds the nearest neighbours to each point and draw lines between each point and its neighbours as shown in the following code.

Software four – This software processes a set of images, convert it to grayscale then takes the bold scale of the grayscale which is between (1-110) then discards other points, makes data as an array, finally the nearest neighbours to each point is searched and draw lines between points as shown in the following code.

Software five – This software takes one image at an execution but with high grayscale points.

4.2 RFFNN Results of the Application

The parameters after that is extracted from each software as inputs to RFFNN, recurrent parameters are affected to train the network for predicting of the effort of new software. Table (2) shows the results of RFFNN of the application. This table refers to recurrent stages and the inputs which include additional inputs and final outputs of the RFFNN. The value (0.6) in the last line of the table refers to the software effort of the future software this is a predicted value depends on the previous software which are of the same specifications in previous times. Table (3) refers to linguistic values of the RFFNN inputs and outputs. Where the linguistic value of the predicted value of software effort of the fifth software (0.6) in the last line of the table is (above average). The RFFNN is evaluated using mean square error (MSE) as shown in equation (5), where the results of the error using MSE equation is a very small value close to zero.

$$MSE = \left[\frac{1}{2}\sum_{i=1}^{n}(P_i - A_i)^2\right] \quad (5)$$

Where,

P_i: estimated value for data output i.

Ai: actual value for the output i.

n: the total number of output

Table2: Results of RFFNN of the application0

Stage	Inputs													Outputs
	Recurrent parameters	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	Output(SE)
1	0	1	1	3	5	1	1	2	1	1	1	1	1	0.4
2	0 0.4	2	2	3	5	1	3	2	1	1	1	1	1	0.5
3	0 0.4 0.5	4	2	3	5	1	3	2	1	1	2	1	1	0.6
4	0 0.4 0.5 0.6	5	2	4	5	1	4	2	1	1	2	1	1	0.6
5	0 0.4 0.5 0.6 0.6	5	2	3	5	1	4	2	1	1	2	1	1	0.6
Where, P1-P12 is the twelve parameters from Size to Documentation.														

Table 3: Linguistic values of the RFFNN inputs and outputs of the application

Stage	Inputs												Output
	Rec.	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	SE
1	0	Very Low	Very Low	Medium	Operations	One	Very simple	MATLAB	Available	Structural	Able	Able	Able	Below average **0.4**
2	0 0.4	Low	Low	Medium	Operations	One	Medium	MATLAB	Available	Structural	Able	Able	Able	Average **0.5**
3	0 0.4 0.5	High	Low	Medium	Operations	One	Medium	MATLAB	Available	Structural	Semi-able	Able	Able	Above average **0.6**
4	0 0.4 0.5 0.6	Very High	Low	Old	Operations	One	Medium	MATLAB	Available	Structural	Semi-able	Able	Able	Above average **0.6**
5	0 0.4 0.5 0.6 0.6	Very High	Low	Medium	Operations	One	Medium	MATLAB	Available	Structural	Semi-able	Able	Able	Above average **0.6**
Where, P1-P12 is the twelve parameters from Size to Documentation.														

5. CONCLUSION

In this paper prediction of software effort is discussed which is one of software engineering techniques to avoid exceeding the limits of the available budget and to determine the involving of functions across the process. Therefore it must be done in the first stages of any software design. Therefore, novel methods for software effort prediction by implementing feed-forward neural networks had been proposed. Since RFFNN is an effective type of feed-forward neural networks and through it the best approximation between the desired and actual output is obtained. This RFFNN is improved from ordinary FFNN to FFNN in a recurrent fashion to give the network the ability of prediction of future software effort not only the effort estimation of the current software. Software effort prediction process has a significant role in the performance of software engineering, because of the performance parameters such as execution time, software size, hardware ability, application type, programming team, complexity, programming language, programming type, development ability, debugging ability, and documentation. They are affected proportionally when the RFFNN used for software effort prediction. In the proposed RFFNN, noticed that the prediction of future software effort is done dependent on similar software efforts in previous stages. Also when the number of previous software increased the prediction become more accurate. RFFNN used the principle of time series to make predictions where the output of time (t) return as input in time (t-1) this could use for developing the feed-forward neural network to feed-forward neural network with a recurrent fashion. The following works are suggested to be future works, RFFNN could be used in other software engineering fields such as predicting for costs in building projects or the size of any software system. As a future work, another type of intelligent methods could be used with Neural Networks and also enhanced such as hybrid systems like the Neuro-Fuzzy system and also, RFFNN could be optimized by using genetic algorithms. RFFNN could use the traditional methods to compute the desired output to network output, therefore a traditional method as COCOMO could be enhanced and used for computing the desired output.

6. REFERENCES

[1] I. Marsic, Software Engineering. Rutgers University, 2012.

[2] I. Sommerville, Software Engineering. Pearson, 2010.

[3] S. Waghmode and K. Kolhe, "A Novel Way of Cost Estimation in Software Project Development Based on Clustering Techniques," Int. J. Innov. Res. Comput. Commun. Eng., vol. 2, no. 4, pp. 3892–3899, 2014.

[4] H. Hamza, a Kamel, and K. Shams, "Software Effort Estimation Using Artificial Neural Networks: A Survey of the Current Practices," Inf. Technol. New Gener. (ITNG), 2013 Tenth Int. Conf., pp. 731–733, 2013.

[5] H. Karna and S. Gotovac, "Estimating software development effort using Bayesian networks," pp. 229–233, 2015.

[6] P. Kaur and R. Singh, "A Proposed Framework for Software Effort Estimation Using the Combinational Approach of Fuzzy Logic and Neural Networks," Int. J. Hybrid Inf. Technol., vol. 8, no. 10, pp. 73–80, 2015.

[7] A. Idri, A. Hassani, and A. Abran, "RBFN Networks-based Models for Estimating Software Development Effort: A Cross-validation Study," 2015 IEEE Symp. Ser. Comput. Intell., vol. 39, no. Ml, pp. 976–983, 2015.

[1] [8] M. T. Hagan, H. B. Demuth, and M. H. Beale, "Neural Network Design," Bost. Massachusetts PWS, vol. 2, p. 734, 1995.

[9] B. Krose and P. Van Der Smagt, An Introduction to Neural Networks, no. University of Amsterdam. 1996.

[10] M. Kubat, "Neural networks: a comprehensive foundation by Simon Haykin, Macmillan, 1994, ISBN 0-02-352781-7.," The Knowledge Engineering Review, vol. 13, no. 4. pp. 409–412, 1999.

[11] Laurene V. Fausett, Fundamentals of Neural Networks. Pearson Education, 1994.

[12] R. C. Lyne and J. C. Maximino, "An Artificial Neural Network Approach to Structural Cost Estimation of Building Projects in the Philippines," Present. DLSU Res. Congr. 2014, vol. 3, pp. 1–8, 2014.

[13] S. Abbinaya and M. S. Kumar, "Software effort and risk assessment using decision table trained by neural networks," 2015 Int. Conf. Commun. Signal Process., pp. 1389–1394, 2015.

[14] P. Agrawal and S. Kumar, "Early Phase Software Effort Estimation Model," 2016 Symp. Colossal Data Anal. Netw., 2016.

[15] M. Bisi and N. K. Goyal, "Software development efforts prediction using artificial neural network," IET Inst. Eng. Technol., vol. 10, no. 3, pp. 63–71, 2016.