IMPROVING MODIFIED COCOMO II ARTIFICIAL NEURAL NETWORK USING HYPERBOLIC TANGENT ACTIVATION FUNCTION

SARAH ABDULKAREM ABDULAZIZ AL-SHALIF

A thesis submitted in
fulfillment of the requirement for the award of the
Degree of Master of Information Technology

Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia

December, 2017

DEDICATION

"In the name of Allah, The Most Gracious, Most Merciful".
Special dedication to my family, to my husband, Abdullah Faisal Alshalif.
To my supervisor, Dr. Noraini Binti Ibrahim. To my friends.
Hearty thanks for the love, support, motivation, encouragement throughout this journey this dissertation is dedicated to all of you.

ACKNOWLEDGEMENT

First and foremost, all the thanks and blessing to the almighty Allah for the strength to complete this journey. At this moment, I would like to express my special gratitude to my precious supervisor, Dr. Noraini Binti Ibrahim for her guidance, patience, advises, ideas, knowledge and vitality till the end of the voyage. I would like to thank the Ministry of Higher Education for sponsoring this research through Fundamental Research Grant Scheme (FRGS) vote number 1610 and thank to the Centre for Graduate Studies (CGS) and the Office of Research, Innovation, Commercialization and Consultancy (ORICC) for the support towards my postgraduate studies.

I am obligated to my dearest family for the endless support, prayer and love. My special appreciation to all my friends. My acknowledgment also goes to the Faculty of Computer Science and Information Technology for the cooperation and accommodations. The least, it is my pleasure to thank to whoever has helped me either directly or indirectly. I would not be able to accomplish this work without the help and support from everyone related in various ways. Thanks.

ABSTRACT

Software cost estimation is a complex and critical issue in software industry but it is an inevitable activity in the software development process. It is one of important factors for projects failure due to the ambiguity and uncertainty of software attributes at the early stages of software development. The estimation of effort in COCOMO II depends on several software attributes namely software size (SS), scale factors (SFs) and effort multipliers (EMs). Several researchers integrate COCOMO II with Artificial Neural Network (ANN) to overcome the ambiguous and uncertain of these attributes. However, ANN contributes to slow convergence caused by sigmoid function. Thus, this research proposes Hyperbolic Tangent activation function (Tanh) to be used in the hidden layer of the ANN architecture to produce faster convergence. Back-propagation learning algorithm is applied to the multilayer neural network for training and testing. The proposed activation function has been trained and tested using two different architectures of NN which are basic COCOMO II-NN and modified COCOMO II-NN that uses COCOMO II NASA93 dataset. The result has been compared to different activation functions namely Uni-polar sigmoid, Bi-polar sigmoid, Gaussian and Softsign. The experiment results indicate that Tanh with modified COCOMO II-NN architecture achieved 23.2780 % Mean Magnitude Relative Error (MMRE) for 19 testing projects and 9.8948 % MMRE for 9 testing projects which is the lowest MMRE among other activation functions. In conclusion, Tanh with modified architecture of COCOMO II-NN provides much better estimation results than other methods and can lead to improvement of software estimates.

ABSTRAK

Membuat anggaran kos perisian adalah suatu aktiviti yang kompleks dan kritikal dalam industri perisian, namun ia merupakan aktiviti yang tidak dapat dielakkan dalam proses pembangunan perisian. Aktiviti ini juga merupakan salah satu faktor penting kegagalan projek kerana atribut perisian yang tidak jelas dan tidak pasti di peringkat awal pembangunan perisian. Anggaran usaha dalam COCOMO II bergantung kepada beberapa atribut iaitu saiz perisian (SS), faktor skala (SFs) dan pengganda usaha (EMs). Beberapa penyelidik mengintegrasikan COCOMO II dengan Rangkaian Neural Buatan (ANN) untuk mengatasi masalah atribut perisian yang kabur dan tidak pasti. Walau bagaimanapun, ANN memperlahankan penumpuan yang disebabkan oleh fungsi sigmoid. Oleh itu, kajian ini mencadangkan fungsi pengaktifan *Tangent Hyperbolic* (Tanh) untuk digunakan dalam lapisan tersembunyi senibina ANN untuk menghasilkan penumpuan yang lebih cepat. Algoritma rambatan balik digunakan untuk rangkaian neural berbilang lapisan untuk latihan dan ujian. Fungsi pengaktifan yang dicadangkan telah dilatih dan diuji menggunakan dua senibina rangkaian neural yang berlainan iaitu COCOMO II-NN asas dan COCOMO II-NN diubahsuai serta menggunakan COCOMO II NASA93 set data. Hasilnya telah dibandingkan dengan fungsi pengaktifan yang berbeza iaitu sigmoid Uni-polar, sigmoid Bi-polar, Gaussian dan Softsign. Keputusan eksperimen menunjukkan bahawa Tanh dengan senibina COCOMO II-NN diubahsuai mencapai 23.7780% magnitud min ralat nisbi (MMRE) untuk ujian kepada19 projek dan 9.8948% MMRE untuk ujian kepada 9 projek yang merupakan MMRE terendah di antara fungsi pengaktifan lain. Kesimpulannya, Tanh dengan seni bina COCOMO II-NN diubahsuai memberikan hasil anggaran yang lebih baik daripada kaedah lain dan boleh menyumbang kepada anggaran perisian yang lebih baik.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

COCOMO 81	-	Constructive Cost Model I
COCOMO II	-	Constructive Cost Model II
SLIM	-	Software Lifecycle Management
FPA	-	Function Point Analysis
EMs	-	Effort Multipliers
SFs	-	Scale Factors
SLOC	-	Source Line of Code
ANN	-	Artificial Neural Network
Tanh	-	Hyperbolic Tangent Activation Function
BP	-	Back-propagation
MRE	-	Magnitude of Relative Error
MMRE	-	Mean Magnitude of Relative Error
SS	-	Software Size
FP	-	Function Point
ILF	-	Internal Logical Files
ELF	-	External Logical Files
EI	-	External Inputs
EO	-	External Outputs
EQ	-	External inquiries
FLANN	-	Functional link Artificial Neural Network
MLP	-	Multilayer Perceptron

RBFNN	-	Radial Basis Function Neural Network
GRNN	-	General Regression Neural Network
PRED	-	Prediction Level
RMSRE	-	Root Mean Square Relative Error
PSO	-	Particle Swarm Optimization
PCA	-	Principle Component Analysis
b	-	Bias
δ	-	Error correction
α	-	Learning rate
i	-	Number of inputs
n	-	Number of nodes
e	-	Natural logarithms

LIST OF PUBLICATIONS

1. **Alshalif, S. A.,** Ibrahim, N. & Herawan, T. (2016). Artificial Neural Network with Hyperbolic Tangent Activation Function to Improve the Accuracy of COCOMO II Model. In *International Conference on Soft Computing and Data Mining*. Springer, Cham. pp. 81-90. SCOPUS indexed.
2. **Alshalif, S. A**., Ibrahim, N., & Waheeb, W. (2017). Improving the Accuracy of COCOMO II Effort Estimation Based on Neural Network with Hyperbolic Tangent Activation Function. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC), Vol. 9, No. 3-5,* pp. 77-82. SCOPUS indexed.

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

Software cost estimation is the process to estimate the effort required to develop software engineering projects (Lindstrom, 2004). This estimating process is one of the most challenging tasks and complex activities in the area of software engineering and project management. Although software cost estimation may be an easy concept, it is in fact difficult and complicated issue (Jones, 2002). Several software cost estimation models have been proposed and developed such as Boehm's Constructive Cost Model I and II or known as COCOMO 81 and COCOMO II, Expert Judgment (Boehm, Madachy & Steece, 2000a), Albrecht's Function Point Analysis (FPA) (Albrecht, 1979) and Putnam's Software Lifecycle Management (SLIM) (Putnam, 1978) which can be classified into two categories namely algorithmic and non-algorithmic models (Boehm *et al.*, 2000a).

Algorithmic models are established based on statistical analysis of past projects data such as cost drivers with its effort multipliers (EMs) and scale factors (SFs). Algorithmic models are also known as the conventional method that provides mathematical and experimental equations to compute software effort (Strike, El Emam & Madhavji, 2001; Khatibi & Jawawi, 2011). The most popular algorithmic cost estimation models are Constructive Cost Model (COCOMO 81 and COCOMO II), Function Point Analysis and Software Life Cycle Management. Algorithmic models need many specific requirements that are also known as software attributes for examples source line of code (SLOC), cost drivers, scale factors, number of user screen and interfaces. Software attributes are difficult to gain at the early stages of software development. Non-algorithmic models published in 1990s such as expert judgment, price-to-win, and machine learning approaches (Boehm *et al.*, 1995; Boehm *et al.*, 2000a). Non-algorithmic models provide powerful linguistic representation that helps to represent more accurate software attributes and can overcome algorithmic models defects when combined with other methods such as Fuzzy Logic and Artificial Neural Networks (ANN) (Srinivasan & Fisher, 1995).

COCOMO 81 is one of the most popular algorithmic software cost estimation model proposed in 1981 by Barry Boehm. It is one of the most well-known and widely used algorithmic cost estimation models in the 1980s. In 1990s, COCOMO 81 faced many problems and difficulties in term of software estimation that are developed using new lifecycle processes approaches, for instance, rapid development and object-oriented approaches. Therefore, to avoid these problems, Boehm improved and published the latest version of COCOMO 81 that is COCOMO II in 1995 (Boehm *et al.*, 1995).

COCOMO II is a model that thinks about the effort needed for software development (Boehm *et al.*, 2000a). It provides accurate effort estimates for both current and likely future software projects (Boehm, Abts & Chulani, 2000b). It involves three sublevels which are Application-Composition model, Early Design model, and Post-Architecture model. The Application-Composition model supports the earliest phases or spiral cycles involved in prototyping the activities that occur in the SDLC. The Early Design model is a high-level model that supports the next phase or spiral cycles that involves alternatives for exploring architecture or strategies for incremental development. Post-Architecture model is suitable for projects that are ready to be developed and it is a more detailed and widely used model (Boehm *et al.*, 2000a).

1.1 Research Motivation

Accurate software cost estimation is highly required in software project management (Boehm *et al.*, 2000a). The software cost estimation is very critical in software engineering and it is an important factor for project failures. This reason motivates the researchers to conduct research on software estimation for better estimations (Lynch, 2009). Accurate software estimates at the early phase of software development is one of the crucial objectives in software project management because of the ambiguity and uncertainty of software attributes due to the difficulty to obtain these attributes at the early stages of the software development (Boehm, 1981).

COCOMO II issoftware cost estimation model developed to improve estimation accuracy (Boehm *et al.*, 2000a). Several research attempts to enhance the existing COCOMO II model to produce better estimation accuracy by incorporating the model with other techniques such as soft computing techniques. One of the most popular soft computing techniques is ANN. Many researchers show that COCOMO II produces more accurate results while incorporated with ANN and can overcome the ambiguity and uncertainty of the software attributes (Kaushik Soni & Soni., 2013; Dan, 2013; Attarzadeh & Ow, 2014; Sarno *et al*., 2015a; Rijwani & Jain, 2016; Strba *et al.*, 2017).

In a broad sense, a neural network structure is usually developed to match the present problem. Many network architectures have been developed for various applications. The performance of a neural network relies on the architecture and their parameters. There are many parameters controlling the architecture of the neural network including number of layers, number of nodes in each layer, activation function in each node, learning algorithm and weights which determine the connectivity between nodes. There is no standard for a perfect parameter in neural network, even small changes of the parameter can cause major variations in the network performance (Senyard, Kazmierczak & Sterling, 2003). In COCOMO II-NN, there are two different architectures for Multilayer Perceptron (MLP). They are basic COCOMO II-NN and modified COCOMO II-NN (Sarno *et al*., 2015a).

Reddy & Raju (2009) proposed and improved the accuracy of COCOMO II using neural network with identity function and modified COCOMO II-NN. While, Kaushik *et al.* (2013) used the same approach with sigmoid activation function. But, the sigmoid function caused slow convergence for the Back-propagation (BP) learning algorithm of ANN (Segee, 1993). While, Bishap (1995) stated that the use of Tanh produced faster convergence of the learning algorithms compared to sigmoid function and the use of Tanh is more efficient for the performance of BP learning algorithm.

Another study was conducted by Poonam and Sonal (2016) to improve COCOMO II model using the basic COCOMO II-NN with Hyperbolic Tangent activation function (Tanh). On the other hand, Sarno *et al*. (2015a) claimed that the modified COCOMO II-NN is more accurate than the basic COCOMO II-NN using sigmoid function. Thus, this research is inspired from the architecture developed by Kaushik *et al*. (2013) and explores the impact and usability of Tanh on modified

COCOMO II-NN to overcome the limitation of sigmoid function. This research accommodates the COCOMO II Post-Architecture model using modified COCOMO II-NN with Hyperbolic Tangent activation function (Tanh) in its hidden layer.

1.2 Aims and Objectives

The aim of this research is to improve the accuracy of COCOMO II and develop a model, namely COCOMO II-NN-Tanh for estimating software effort by incorporating COCOMO II and ANN with Tanh activation function. The study embarks on the following objectives:

I. To propose and develop COCOMO II-NN-Tanh based on BP learning algorithm.
II. To evaluate and compare the results of COCOMO II-NN-Tanh with other models that use other activation functions namely Uni-polar sigmoid, Bi-polar sigmoid, Gaussian and Softsign.

1.3 Scope of the Study

This research integrated COCOMO II and ANN with Tanh. This research focused on the Post-Architecture model of COCOMO II with BP learning algorithm. Two different architectures of ANN ware used. They are basic and modified COCOMO II-NN.

Tanh has good performance for the prediction problems in the modified COCOMO II-NN. Nonetheless, Tanh has limitation on the basic architecture due to the characteristics of the basic COCOMO II-NN when used gaussian function. MLP that used gaussian function and has three layers is called Radial Basic Neural Network (RBNN). The main advantages of the RBNN over other MLP networks include fast learning (Lee et al., 1999), easy design, good generalization, strong tolerance to input noise and online learning ability (Yu et al., 2011). Thus, these contributions show that the gaussian function more suitable for the basic COCOMO II-NN and Tanh is more suitable for the modified COCOMO II-NN architecture.

Besides, COCOMO II NASA93 dataset was used for training and testing processes which has 93 projects that are available on the public domain from TERA

(2016). The entire COCOMO II NASA93 dataset was attached in APPENDIX A. The training, testing, implementation and calculation for this study were done using MATLAB (R2013a).

1.4 Significance of the Study

This study mainly concerned on how to improve the estimation accuracy of COCOMO II because as mentioned earlier, this model can be helpful in predicting the effort needed to develop a software. At the end of this research, it is found that the proposed method produced better result that is closer to the actual effort compared to the original COCOMO II. This implies that the proposed method managed to improve the estimation accuracy of COCOMO II.

Inaccurate estimation is the main reason for projects failure and may cause projects to be terminated (Molokken & Jorgensen, 2003). As such, accuracy is a very important issue in software cost estimation especially for executives, managers, technical staff and particularly practitioners who carry out or depend on cost estimation (Kemerer, 1987). Accurate estimation of software development cost remains a challenge for software engineering research due to the shortcomings and inaccuracies of the models (MacDonell & Gray, 1997).

1.5 Thesis Outline

This chapter had described the research motivation, aims and objectives, scope of the research, as well as the structure of the whole thesis. The rest of the thesis was organized as follows.

Chapter 2 focuses on the overview of the software cost estimation, as well as algorithmic models. The discussion then continues with the description of activation functions, evaluation criteria and comparison made between of the existing techniques.

Chapter 3 illustrates the research methodology and discusses on two main steps of this research which include COCOMO II-NN-Tanh processes and evaluation and comparison of the developed model with other models.

Chapter 4 The implementation of ANN with COCOMO II is discuss. Then discussion on the dataset. The results then compare with two different architectures namely basic COCOMO II-NN and modified COCOMO II-NN. Results obtain will analyze based on Magnitude of Relative Error (MRE) and Mean Magnitude of Relative Error (MMRE) evaluation criteria.

Finally, **Chapter 5** concludes the thesis and presents several recommendations for future work.

CHAPTER 2

LITERATURE REVIEW

This chapter presents a report of studies found in the literature related to the research. Section 2.1 described the terms used in this research in detail and explored the main topics of the research including software cost estimation models. Section 2.2 discussed on the activation functions and their different types. A general overview of the evaluation criteria was discussed in Section 2.3. Section 2.4 discussion of the analysis of variance (ANOVA). A systematic literature review is presented in Section 2.5. Section 2.6 summarized the whole chapter.

2.1 Software Cost Estimation

Software cost estimation is the process to estimate and predict the effort needed to develop a software. It is the most crucial and challenging task in software project management in specific and software engineering field in general (Jones, 2002). Software project managers and developers are interested to accurately estimate the effort needed at the early stage of software development. This process estimates the effort required to develop software based on the software attributes. The degree of accuracy is measured by comparing the effort obtained from the estimation process with the actual effort obtained from the dataset. The estimation is considered more accurate when the estimated effort is closest to the actual effort. Several software cost estimation models have been developed and improved (Boehm, 1981; Albrecht, 1979; Putnam, 1978; Patil *et al.*, 2014). The software cost estimation models are divided into two major categories which include algorithmic and non-algorithmic models (Boehm, 1981). The algorithmic models have many advantages although it is difficult to learn models and needs large data for learning. Algorithmic models use mathematical formula to estimate project effort based on the project size and other software attributes (Boehm, 1981). The most popular algorithmic cost estimation models are Boehm’s Constructive Cost Model (COCOMO 81 and COCOMO II), Albrecht’s Function Point Analysis (Albrecht, 1979) and Putnam’s Software Lifecycle Management (Putnam, 1978). On the other hand, non-algorithmic models are easy to learn but require complete information on one very similar previous project to be compared with current software project (Bardsiri *et al.*, 2012). Non-algorithmic models were established based on heuristic approaches and experts’ knowledge. Expert Judgement and Top-Down models belong to this category. The limitations of algorithmic models lead to the exploration of non-algorithmic models which are soft computing based. Non-algorithmic models need to have the knowledge of a previously completed project that is similar to the current software project. Estimation is done on the basis of analysis of previous software projects or datasets. Some of the techniques based on non-algorithmic models for cost estimation are artificial neural networks (ANNs) and fuzzy logic (FL) (Shekhar & Kumar, 2016). Thus, it was observed by Shekhar and Kumar (2016) that the integration of these two categories produced more efficient and accurate estimation models.

2.1.1 COCOMO II

COCOMO II is the latest version of COCOMO 81 developed by Boehm in 1995 and it is the most popular and well-known cost estimation model (Tailor, Saini & Rijwani, 2014) due to its flexibility and simplicity for estimating the effort expressed in terms of person-months (PM). PM refered to amount of time one person spends working on the software development project for one month. COCOMO II is used to estimate project effort required to develop a software project. Boehm *et al.* (2000a) categorizes the entire COCOMO II model into three sublevels or models. They are:

- Application-Composition model: This model is suitable for quickly developed applications using interoperable components like components based on graphical user interface builders and is based on new object point’s estimation.
- Early Design model: This model is used in the early stages of a software project and can be used in Application Generator, System Integration or Infrastructure Development Sector. It uses Unadjusted Function Points as the measure of size.

- Post-Architecture model: This is the most detailed of the three and is used after the overall architecture for the project has been designed and could use either function points or line of code as size estimates.

In COCOMO II, there are several fundamental software attributes used to estimate the effort required for developing a system which are software size (SS), cost drivers with its Effort Multipliers (EMs) and scale factors (SFs) (Boehm *et al.*, 2000a). COCOMO II Post-Architecture model and its software attributes is defined by Equation 2.1.

$$Effort_{PM} = A \times SS^{E} \times \Pi_{i=1}^{17} EM_{i}$$

(2.1)

Where

$A = 2.94$

$E = 1.01 + \sum_{j=1}^{5} SF_j$

For Post-Architecture model, there are 17 cost drivers with its EMs, 5 SFs and 1 SS as shown in Figure 2.1

REFERENCES

Albrecht, A. J. (1979). Measuring application development productivity. *Proc. of the Joint SHARE/GUIDE/IBM Applicaiton Development Symposium*. Vol. 10, pp. 83-92.

Amit, D. J. (1992). *Modeling brain function: The world of attractor neural networks*. Cambridge University Press.

ANOVA, O. W. (2008). Analysis of Variance (ANOVA). *Group*, *1*(4), p. 3.

Attarzadeh, I. & Ow, S. H. (2010a). Proposing a New Software Cost Estimation Model Based on Artificial Neural Networks. *IEEE International Conference on Computer Engineering and Technology*. IEEE. pp. V3-487 – V3-491.

Attarzadeh, I. & Ow, S. H. (2010b). Improving the accuracy of software cost estimation model based on a new fuzzy logic model. *World applied sciences Journal 8(2)*. pp. 177-184.

Attarzadeh, I., & Ow, S. H. (2011). Improving estimation accuracy of the COCOMO II using an adaptive fuzzy logic model. In *Fuzzy Systems (FUZZ), 2011 IEEE International Conference*. IEEE. pp. 2458-2464.

Attarzadeh, I. & Ow, S. H. (2014). Proposing an Effective Artificial Neural Network Architecture to Improve the Precision of Software Cost Estimation Model. *International Journal of Software Engineering and Knowledge Engineering*, *24(06)*, pp. 935-953.

Badjate, S. K. & Gaikwad, U. K. (2015). Develop Hybrid Cost Estimation Model For Software Applications.

Bardsiri, V. K., Jawawi, D. N. A., Hashim, S. Z. M. & Khatibi, E. (2012). Increasing the accuracy of software development effort estimation using projects clustering. *IET software*, *6(6)*, pp. 461-473.

Bergstra, J., Desjardins, G., Lamblin, P. & Bengio, Y. (2009). Quadratic polynomials learn better image features. Technical Report, *Department of Computer Science and Operational Research*, *University of Montreal*.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford university press.

Boehm, B. W. (1981). *Software engineering economics*. Englewood Cliffs (NJ): Prentice-hall.

Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R. & Selby, R. (1995). Cost models for future software life cycle processes: COCOMO 2.0. *Annals of software engineering*, *1*(1), pp. 57-94.

Boehm, B. W., Madachy, R. & Steece, B. (2000a). *Software cost estimation with Cocomo II with Cdrom.* Prentice Hall PTR.

Boehm, B., Abts, C. & Chulani, S. (2000b). Software development cost estimation approaches—A survey. *Annals of software engineering*, *10*(1-4), pp. 177-205.

Broomhead, D. S. & Lowe, D. (1988). *Radial basis functions, multi-variable functional interpolation and adaptive networks*. Royal signals and radar establishment malvern. united kingdom.

Castellano, G., Fanelli, A. M. & Pelillo, M. (1997). An iterative pruning algorithm for feedforward neural networks. *IEEE Transactions on Neural Networks*, *8*(3), pp. 519-531.

Chiang, Y. M., Chang, L. C. & Chang, F. J. (2004). Comparison of static-feedforward and dynamic-feedback neural networks for rainfall–runoff modeling. *Journal of hydrology*, *290*(3), pp.297-311.

Crichton, N. (2000). Analysis of variance (ANOVA).

Dan, Z. (2013). Improving the accuracy in software effort estimation: Using artificial neural network model based on particle swarm optimization. *Proc. 2013 IEEE International Conference on Service Operations and Logistics, and Informatics*. IEEE . pp. 180-185.

Deeson, E. (1991). *Collins dictionary of information technology*. HarperCollins.

Duch, W. & Jankowski, N. (1999). Survey of neural transfer functions.*Neural Computing Surveys*, *2(1)*, pp. 163-212.

Fidele, B., Cheeneebash, J., Gopaul, A. & Goorah, S. S. (2009). Artificial neural network as a clinical decision-supporting tool to predict cardiovascular disease. *trends in applied sciences research*, *4(1)*, pp. 36-46.

Froelich, A. G., Stephenson, W. R. & Duckworth, W. M. (2008). Assessment of materials for engaging students in statistical discovery. *Journal of Statistics Education*, *16*(2), pp.10-22.

Georgi, R., Vogt, T., Team, O. S. A. & Center, L. B. J. S. (2008). Illustrative Example of a Function Point Analysis for the NASA Crew Exploration Vehicle Guidance, Navigation & Control Flight Software. *National Aeronautics and Space Administration, Johnson Space Center*.

Glorot, X. & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. P*roc. of the Thirteenth International Conference on Artificial Intelligence and Statistics.* pp. 249-256.

Hill, T., Marquez, L., O'Connor, M. & Remus, W. (1994). Artificial neural network models for forecasting and decision making. *International Journal of Forecasting, 10(1),* pp. 5-15.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A. R., Jaitly, N. & Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, *29*(6), pp. 82-97.

Hornik, K., Stinchcombe, M. & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, *2(5)*, pp. 359-366.

Hussain, H. M. (2014). *Linguistic Approaches for Early Measurement of Functional Size from Software Requirements.* Concordia University: Doctoral dissertation.

IBM knowledge center (2005). from http://www.ibm.com/support/knowledgecenter/SSRTLW_6.0.1/com.ibm.rational.test.ct.doc/topics/c_datapartitioning.html

Ishak, I. S. & Alias, R. (2005). Designing a strategic information systems planning methodology for malaysian institutes of higher learning .*Issues in Information Systems*, *6*(1), pp.325-331.

Jones, C. (2002). Software cost estimation in 2002. *The Journal of Defense Software Engineering*, *15(6)*, pp. 4-8.

Kaastra, I. & Boyd, M. (1996). Designing a neural network for forecasting financial and economic time series. *Neurocomputing*, *10(3)*, pp. 215-236.

Karlik, B. & Olgac, A. V. (2011). Performance analysis of various activation functions in generalized MLP architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, *1*(4), pp. 111-122.

Kaushik, A., Chauhan, A., Mittal, D. & Gupta, S. (2012). COCOMO Estimates Using Neural Networks. *International Journal of Intelligent Systems and Applications*, *4*(9), pp. 22.

Kaushik, A., Soni, A. K. & Soni, R. (2013). A Simple Neural Network Approach to Software Cost Estimation. *Global Journal of Computer Science and Technology*

Kaushik, A., Tayal, D. K., Yadav, K. & Kaur, A. (2016). Integrating firefly algorithm in artificial neural network models for accurate software cost predictions. *Journal of Software: Evolution and Process*, *28*(8), pp. 665-688.

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

Khatibi, V. & Jawawi, D. N. (2011). Software cost estimation methods: A review 1.

Kotsiantis, S. B., Kanellopoulos, D. & Pintelas, P. E. (2006). Data preprocessing for supervised leaning. *International Journal of Computer Science*, *1(2)*, pp. 111-117.

Kumar, G. & Bhatia, P. K. (2014). Automation of software cost estimation using neural network technique. *International Journal of Computer Applications, 98(20).*

Larochelle, H., Bengio, Y., Louradour, J. & Lamblin, P. (2009). Exploring strategies for training deep neural networks. *The Journal of Machine Learning Research*, *10*, pp. 1-40.

Lindstrom, B. (2004). A software measurement case study using GQM. *J. Lund Univ., USA*.

Lo, B. & Gao, X. (1997). Assessing software cost estimation models: criteria for accuracy, consistency and regression. *Australasian Journal of Information Systems*, *5*(1).

Lodwich, A., Rangoni, Y., & Breuel, T. (2009). Evaluation of robustness and performance of early stopping rules with multi layer perceptrons. *Proc. International Joint Conference on Neural Networks.* IEEE. pp. 1877-1884.

Lynch, J. (2009). Chaos manifesto. The Standish Group, Boston.

Madheswaran, M. & Sivakumar, D. (2014). Enhancement of prediction accuracy in COCOMO model for software project using neural network. *Proc. International Conference on Computing, Communication and Networking Technologies.* IEEE. pp. 1-5.

MacDonell, SG. & Gray, AR. (1998). A comparison of modeling techniques for software development effort prediction.

Markopoulos, A. P., Georgiopoulos, S. & Manolakos, D. E. (2016). On the use of back propagation and radial basis function neural networks in surface roughness prediction. *Journal of Industrial Engineering International*, *12*(3), pp.389-400.

Mehtani, P. & Priya, A. (2011). *Pattern Classification using Artificial Neural Networks*. Doctoral dissertation.

Meli, R. & Santillo, L. (1999) . Function point estimation methods: A comparative overview. *Proc. FESMA*. pp. 6-8.

Moller, M. F. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks*, *6*(4), pp. 525-533.

Molokken, K. & Jorgensen, M. (2003). A review of software surveys on software effort estimation. *Proc. 2003 International Symposium on Empirical Software Engineering.* IEEE. pp. 223-230.

Mortimer, R. G. (1999). *Mathematics for physical chemistry*. Academic Press.

Mukherjee, S. & Malu, R. K. (2014). Optimization of project effort estimate using neural network. P*roc. 2014 International Conference on Advanced Communication Control and Computing Technologies*. IEEE. pp. 406-410.

Nassif, A. B. (2012). *Software size and effort estimation from use case diagrams using regression and soft computing models*. The University of Western Ontario: Doctoral dissertation.

Nawi, N. M., Atomi, W. H. & Rehman, M. Z. (2013). The effect of data pre-processing on optimized training of artificial neural networks. *Procedia Technology*, *11*, pp. 32-39.

Patil, L. V., Waghmode, R. M., Joshi, S. D. & Khanna, V. (2014). Generic model of software cost estimation: A hybrid approach. *2014 IEEE International*

Advance Computing Conference.IEEE. pp. 1379-1384.

Putnam, L. H. (1978). A general empirical solution to the macro software sizing and estimating problem. *IEEE transactions on Software Engineering*, *4(4),* pp. 345-361.

Putnam, L. H. & Myers, W. (1991). *Measures for excellence: reliable software on time, within budget*. Prentice Hall Professional Technical Reference.

Rao, B. T., Sameet, B., Swathi, G. K., Gupta, K. V., RaviTeja, C. & Sumana, S. (2009). A novel neural network approach for software cost estimation using Functional Link Artificial Neural Network (FLANN). *International Journal of Computer Science and Network Security*, *9*(6), pp. 126-131.

Reddy, C. S. & Raju, KVSVN. (2009). A concise neural network model for estimating software effort. *International Journal of Recent Trends in Engineering*, *1(1)*, pp. 188-193.

Rijwani, P., & Jain, S. (2016). Enhanced Software Effort Estimation Using Multi Layered Feed Forward Artificial Neural Network Technique. *Procedia Computer Science*, *89*, pp. 307-312.

Rojas, R. (2013). *Neural networks: a systematic introduction*. Springer Science & Business Media.

Saen, R. F. (2009). The use of artificial neural networks for technology selection in the presence of both continuous and categorical data. *World Applied Sciences Journal*, *6*(9), pp. 1177-1189.

Sallehuddin, R., Shamsuddin, S. M. H. & Hashim, S. Z. M. (2008). Application of grey relational analysis for multivariate time series. *Eighth International Conference on Intelligent Systems Design and Applications, 2008.* IEEE. pp. 432-437.

Sarno, R., Sidabutar, J. & Sarwosri (2015a). Comparison of different Neural Network architectures for software cost estimation. *Proc. International Conference on Computer, Control, Informatics and its Applications.* IEEE. pp. 68-73.

Sarno, R., Sidabutar, J. & Sarwosri (2015b). Improving the accuracy of COCOMO's effort estimation based on neural networks and fuzzy logic model. *Proc. 2015 International Conference on Information & Communication*

Technology and Systems. IEEE. pp. 197-202.

Segee, B. E. (1993). Using spectral techniques for improved performance in artificial neural networks. *Proc. of IEEE Int. Conf. on Neural Net.* San Francisco, CA, USA.

Senyard, A., Kazmierczak, E. & Sterling, L. (2003). Software engineering methods for neural networks. *Software Engineering Conference*. Tenth Asia-Pacific: IEEE. pp. 468-477.

Shekhar, S. & Kumar, U. (2016). Review of Various Software Cost Estimation Techniques. *International Journal of Computer Applications*, *141*(11).

Shivakumar, N., Balaji, N. & Ananthakumar, K. (2016). A Neuro Fuzzy Algorithm to Compute Software Effort Estimation. *Global Journal of Computer Science and Technology*, *16*(1).

Shores, T. S. (2007). *Applied linear algebra and matrix analysis*. Springer Science & Business Media.

Sibi, P., Jones, S. A. & Siddarth, P. (2013). Analysis of different activation functions using back propagation neural networks. *Journal of Theoretical and Applied Information Technology*, *47*(3), pp. 1264-1268.

Singh, S. & Kaur, R. (2015). Estimating effort for corrective software maintenance using neural networks and regression technique. *International Journal of Computer Science and Communication Engineering, 4(1),* pp. 39-44.

Sivakumar, D., & Janaki, K. (2017). Enhancing the Software Effort Prediction Accuracy Using Reduced Number of Cost Estimation Factors with Modified COCOMO II Model.

Socher, R. & Mundra, R. S. (2016). CS 224D: Deep Learning for NLP1.

Specht, D. F. (1991). A general regression neural network. *IEEE Transactions on Neural Networks, 2*(6), pp. 568-576.

Srinivasan, K. & Fisher, D. (1995). Machine learning approaches to estimating software development effort. *IEEE Transactions on Software Engineering* , *21*(2), pp. 126-137.

Stergiou, C. & Siganos, D. (2010). Neural Networks.

Strba, R., Stolfa, S., Stolfa, J., Vondrak, I. & Snasel, V. (2017). An Application of Neural Network in Method for Use Case Based Effort

Estimation. *Information Modelling and Knowledge Bases XXVIII*, *292*, pp. 231.

Strike, K., El Emam, K. & Madhavji, N. (2001). Software cost estimation with incomplete data. *IEEE Transactions on Software Engineering*, *27*(10), pp. 890-908.

Tailor, O., Saini, J. & Rijwani, M. P. (2014). Comparative Analysis of Software Cost and Effort Estimation Methods: A Review. *Interfaces*, *5(7)*, pp.10.

TERA. (2016). from http://openscience.us/repo/effort/cocomo/.

Todhunter, C. (2003). Analysis of variance (ANOVA). *The AZ of Social Research: A Dictionary of Key Social Science Research Concepts*, pp. 9.

Tolue, S. F. & Akbarzadeh-T, M. R. (2013). Dynamic fuzzy learning rate in a self-evolving interval type-2 TSK fuzzy neural network. *Proc. 2013 13th Iranian Conference on Fuzzy Systems.* IEEE. pp. 1-6.

Tu, J. V. (1996). Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. *Journal of clinical epidemiology*, *49*(11), pp. 1225-1231.

Wang, S. C. (2003). Artificial neural network. In *Interdisciplinary computing in java programming*. Springer US. pp. 81-100.

Weisstein, E. W. (2002). Gaussian function. *Wolfram Research, Inc.*

Wilamowski, M. & Jaeger, R. C. (1996). Implementation of RBF type networks by MLP networks. *in Proc. IEEE Int. Conf. Neural Network., Washington.*pp. 1670–1675.

Yu, H., Xie, T., Paszczynski, S. & Wilamowski, B. M. (2011). Advantages of radial basis function networks for dynamic system design. *IEEE Transactions on Industrial Electronics*, *58*(12), pp. 5438-5450.

Zadeh, A. L. (2001). The future of soft computing. *The 9th IFSA World Congress and 20th NAFIPS International Conference*. Vancouver, Canada. pp. 217-228.