Kurdistan Journal of Applied Research (KJAR) | Print-ISSN: 2411-7684 – Electronic-ISSN: 2411-7706 | kjar.spu.edu.iq
Volume 2 | Issue 3 | August 2017 | DOI: 10.24017/science.2017.3.5

Applying Neural Network Approach with Imperialist Competitive Algorithm for Software Reliability Prediction

Shirin Noekhah
Faculty of Computing, Universiti Teknologi of Malaysia, UTM,
81300, Johor, Malaysia
nshirin2@live.utm.my

Naomie binti Salim
Faculty of Computing, Universiti Teknologi of Malaysia, UTM,
81300, Johor, Malaysia
naomie@utm.my

Nor Hawaniah Zakaria
Faculty of Computing, Universiti Teknologi of Malaysia, UTM,
81300, Johor, Malaysia
hawaniah@utm.my

Abstract: ***Software systems exist in different critical domains. Software reliability assessment has become a critical issue due to the variety levels of software complexity. Software reliability, as a sub-branch of software quality, has been exploited to evaluate to what extend the desired software is trustable. To overcome the problem of dependency to human power and time limitation for software reliability prediction, researchers consider soft computing approaches such as Neural Network and Fuzzy Logic. These techniques suffer from some limitations including lack of analyzing mathematical foundations, local minima trapping and convergence problem. This study develops a novel model for software reliability prediction through the combination of Multi-Layer Perceptron Neural Network (MLP) and Imperialist Competitive Algorithm (ICA). The proposed model has solved some of the problems of existing methods such as convergence problem and demanding on huge number of data. This model can be used in complicated software systems. The results prove that both training and testing phases of this model outperform existing approaches in terms of predicting the number of software failures.***

Keywords: Soft computing, reliability of software, Multi-Layer Perceptron Neural Network, Imperialist Competitive Algorithm

1.INTRODUCTION

Recently, computers have become an essential part of human life. They are exploited in widespread range of domains including military, commercial and industrial areas. The critical part of the computer is the software; consequently, using the failure-free software which can complete the desired task with high accuracy and quality is vital. Software reliability as an important factor of software quality applied to assess the system operations profile. According to ANSI (American National Standards Institute) definition, software reliability is the possibility of failure-free software operation evaluated during an accurate time period within an exact environment (, ANSI 1991) [1].

During system analysis, two types of reliability have been considered which include software reliability and hardware reliability. There are significant differences between the software and hardware failures. In fact, if the software is not used, no failure will be occurred and the system will remained reliable. While in hardware, the reliability is threatened by passing the time, even though the hardware is not used. Table 1 illustrates the differences between these two types of reliability and this study focuses on software reliability.

Table 1: Differences between Software Reliability and Hardware Reliability

	Software reliability	**Hardware reliability**
Failure cause	Design defects	Physical defects
Wear-out	any time	After a time period
Repairable system	Periodic checking	Occurred problem
Time dependency and life cycle	Not a function of time	Related to the time
Environmental factors	Effects on program inputs not reliability	The main threat
Interfaces	Conceptual	Visual

Software is an inseparable part of many systems and devices; therefore the failure prevalence of it is widespread. It covers a wide range of devastations from UI problems to the codding errors. There are many factors cause unreliability in software such as human design, coding error, fixing problems which (sometimes) cause new problems to be arisen, running environment changes, misinterpretation of the desired tasks and insufficient testing. In software reliability (in terms of correctness of the software), the expected outcome of the system and the real software output will be compared for the specific environment and condition.

There is a reverse relationship between software failures and its reliability, which means that as the number of faults increase, the software reliability will decrease. Software reliability model should track the changing of reliability by passing the time, so it can be considered as time-series problem. Complexity of

system increases the risk level of reliability in software compared with simple system. So, the developers try to transfer the complexity from the system into the software, and improve the software instead of system whenever it is needed.

Developing a general model which can be exploited in variety domains of study is one of the main challenges in software reliability prediction. To the best of our knowledge, there is no generalized powerful single model applied for all projects and circumstances. The best solution to overcome this problem is proposing the model which does not need any assumption about the environment or external software parameters. In this case, applying soft computing techniques and more specific neural network (NN) is the best solution. Software reliability can be considered as non-linear problem. In this case, artificial neural network is an efficient approach compared with traditional methods due to having the ability of non-linear mapping and using in time-series prediction domain.

On the other hand, inconsistency in data of failures and their frequent changes cause the reliability prediction techniques to be varied in different conditions, and consequently, the traditional prediction models cannot be suitable for them. In addition, the existing method can only predict the time between failures and cannot give the accurate prediction for future. Some methods have been proposed to overcome this limitation by combining the time and test coverage to apply in software reliability prediction. But, since traditional methods are based on simple and shallow assumptions and the relationship between these factors are so complicated, the numerical solution for adjusting parameters of software reliability model is very hard. Among variety NN architectures, most of researchers exploit Multi-Layer Perceptron (MLP) along with variety set of learning algorithms to efficiently perform reliability prediction.

There are variety sets of learning algorithms exploited for training the NN. BP (Back Propagation algorithm), GA (Genetic algorithm) and various Evolutionary algorithms are the most popular NN learning algorithms. In BP, achieving full sampling of the model is hard. This algorithm also suffers from trapping in local minima and low convergence rate. On the other hand, PSO faces other limitations including no solid analysis mathematical foundation, limitation in real time applications and initializing parameter problem and determining the best solution. The aforementioned weaknesses give motivation to use a more powerful evolutionary algorithm (ICA) to train MLP network and overcome most of these problems. It uses sufficient parameters set and mathematical foundation. Due to the robust structure, this algorithm avoids local minima trapping and presents the most effective solution.

In the current study, a novel model has been proposed to forecast the reliability of software by combining the MLP NN and ICA algorithm to achieve high accuracy in terms of predicting the large amount of failures data. It would improve the conventional Software Reliability prediction techniques with some intelligent solutions to deal with unpredictable software system behaviors. Hence, it can maximize working with no failure chance occurred within a specific period of time. Moreover, this study tries to evaluate the various strategies proposed by previous works on software reliability prediction.

2. LITERATURE REVIEW

Neural Network-based models can be exploited generally for various sets of non-linear applications with respect of their desired accuracy. Karunanithi et al. [2] proposed the first NN model of software reliability prediction. Two other models were suggested by Adnan et al. [3] and Park et al. [4] based on using neural networks, and their results showed the effectiveness of their approach compared with analytical models. In existing techniques researchers exploit single-input single-output neural network structure to develop the software reliability models. Cumulative execution time as the input and the number of failures as the output have been considered by Karunanithi et al., in [2]. On the other hand, [5] set the number of failures as input and the time of failure as the output.

Cai et al., exploited the recent 50 times of inter-failure in order to forecasting the next failure time. It showed that the architecture of neural network, i.e. number of hidden layers and neurons in each layer, influences on the network performance [6]. They experimented with variety set of neurons (20-50 neurons), while [3] applied 1-4 neurons as input. In [7] an online adaptive software reliability model has been proposed based on exploiting an evolutionary connectionist approach along with genetic algorithm to enhance the number of neurons. In addition, they modified LM (Levenberg-Marquardt) algorithm with Bayesian regularisation to improve the accuracy of prediction. In another evolutionary NN model, [8] modelled the inter-relationship which exists within software failure data instead of considering the relationship between the time and the data of failures to predict the future failure time.

Jun [9] developed a non-parametric ensemble method based on neural network structure for reliability prediction. He experimented variety sets of predictors and showed the effectiveness of using the combination of predicators compared with applying the single predictor. Jin hybridized GA and SA algorithm (simulated annealing algorithm) to train Support Vector Regression (SVR) model for software reliability prediction. This model decreased the error rate and improved the performance of prediction [10]. As selecting the best SVR's parameters is a very difficult process, variety sets of optimization algorithms have been suggested in terms of finding the most effective combination of parameters. Similarly, Jin optimized EDA (estimation of distribution algorithm) to keep the variety of populations, and proposed IEDA-SVR model that is the hybridization of EDA and SVR model to

improve the selected parameters of SVR. This combination led to improve the performance of software reliability prediction [11]. Park and Baik suggested a novel model which made of the combination of various software reliability models by exploiting decision tree and using multiple criteria to decrease decision tree pruning errors and improve the prediction accuracy [12].

Wu et al. proposed a model by using GRNN (General Regression NN) and assessed the influence of test coverage on predicting the reliability of software to enhance the accuracy. They improved the prediction ability of GRNN during working with small dataset [13]. The efficiency of NN structure compared with analytical model for reliability prediction has been proved by Liu et al. [14]. [15] designed a new model by using the counter and back propagation NNs to estimate the parameters of reliability prediction when the size of dataset is small. Vapnik presented a new neuro-fuzzy technique and the results showed the enhancement of the accuracy of prediction compared with GRNN [16]. RNNBPTT was the new NN structure which proposed by Bhuyan et al. and constructed by the combination of RNN (Recurrent Neural Network) and BP Through Time learning algorithm for predicting software reliability. The results showed that RNN outperforms other existing techniques with consistent behaviors to predict reliability [17]. The ensemble of NNs method was proposed by Bal et al. and the results were compared with existing traditional models by exploiting three standard datasets [18].

In terms of training the NN structure, variety sets of algorithms can be exploited. BP, GA and Evolutionary algorithm are common learning algorithms. Each of them has some weaknesses which give motivation to develop or exploit more powerful algorithm. For example, in BP achieving final model full sampling, local minima trapping and low convergence rate are the main limitation. In addition, PSO suffers from lack of solid analysis mathematical foundation, difficulty of using in real-time application, initializing parameters and the most effective solution finding. Finally, GA not only traps in local minima, but also, suffers from convergence problem. In addition, it has slow execution time and can find the optimal solutions not the exact one.

These problems cause that a novel evolutionary algorithm, ICA has been used which can overcome most of those mentioned problems. Compared with the existing algorithms, ICA solves some of their problems by using adequate parameters and mathematical foundation. Moreover, it avoids trapping in local minima and results the best and optimal solution. Due to all these advantages, ICA has been exploited to train MLP NN and improve the accuracy of reliability prediction.

3. SOFTWARE RELIABILITY PREDICTION METHODOLOGY

Different set of models have been developed to accomplish variety tasks in software engineering domain. Models help developers to have better understanding of existing problem and assign resources more efficient and complete the projects within a desired time period. In addition, model reduces the software design and development's cost and risk. In this study, a novel model has been proposed based on the hybridization of MLP and ICA algorithm.

3.1. Software Reliability Dataset

In supervised learning approach, dataset play a critical role in order to training and testing of proposed model. In evaluation phase, the standard datasets should be applied which are used by most of the researchers. The credibility and efficiency of the prediction model depend on the quality of input dataset. These factors can be assessed based on the dataset nature and collection method. Data can be collected for software reliability prediction by using three popular methods:

- Data produced during testing phase of software.
- Failure data Simulation by using the information of previous projects.
- Users' feedback after releasing the product in the market.

In software reliability domain, there is no specific method for failure data collection. Different research use variety set of software failure datasets, but most of researchers have exploited two standard datasets including DACS (Data and Analysis Centre for Software) and Pham & Pham. We, also, used the same datasets in our study. Size different is the main factor of selecting these datasets to evaluate the proposed model flexibility and reliability for predicting software failures.

Pham & Pham presents a software reliability dataset according to the times of software inner failure. The other dataset is SLED dataset which refers to the Software Life Cycle Empirical/Experience Database presented by DACS. DACS considers the time interval of failures and contains failure data which belong to 16 various projects. From these categories, military dataset (No.40) is exploited which includes 180000 instructions, 101 software failure numbers and collected during system testing phase. Figure 1 plots the relationship between failures' number and time interval between the failures.

For accuracy improvement and increasing the convergence speed of the proposed model, the data should be normalized before sending to the input layer, so that data will be formed in the range of [-1,1]. In addition, the output layer results will be deformalized and converted into the original value. The details of datasets and some samples are presented in [19].

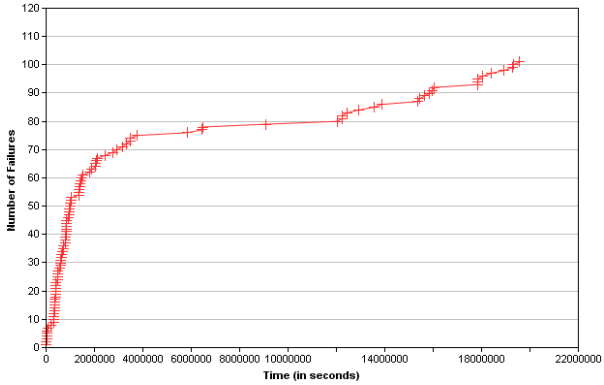

Figure 1 Failure data for project 40

3.2. Software Reliability Framework

The framework of study describes different phases of research methodology and explains how those phases are implemented. In order to satisfy the objectives of research, the methodology is divided into four main phases including analysis, design, implementation and testing and verification. Figure 2 presents the framework of the proposed model framework.

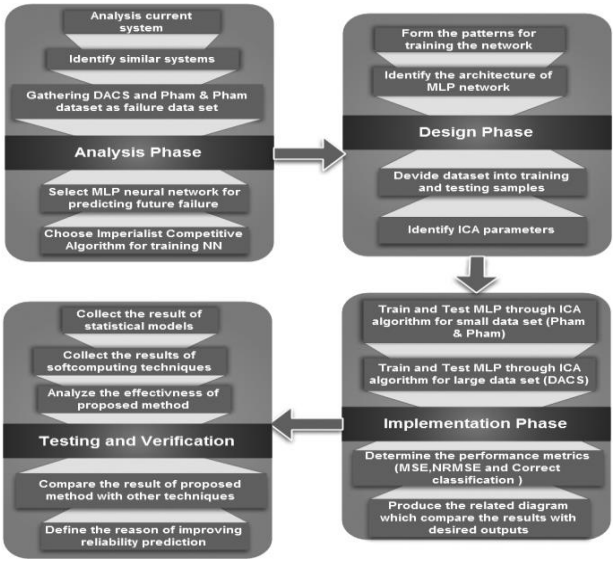

Figure 2 The Framework of MLP-ICA Model

In analysis phase, the current software system will be analyzed and if there is any similar software, it will be identified to benefit of it failure history. The proper dataset is collected in this phase and the most suitable NN structure (in this study MLP) along with the best learning algorithm (ICA) will be exploited. Design phase includes dataset preparation for training and testing of MLP, and also, designing the architecture and layout of MLPNN along with setting the parameters of ICA algorithm. In third phase, the proposed model has been trained and tested with both datasets to evaluate the effect of the characteristics of different types of dataset on the performance of the model. In terms of model assessment, the most accurate performance measurement metrics have been used and the related diagrams, which compare the output of the model and desired output, have been generated. Finally, the accuracy of model will be compared with existing statistical and soft computing techniques.

3.3. ICA-MLP Prediction Model

ICA algorithm and MPL NN have been used to present a hybridized model for software reliability prediction. ICA-MLP as a data-driven model exploits the analysis of time-series (due to the software failure's nature and applied datasets) to identify the future number of failures occurred in the software. ICA algorithm is used as a learning algorithm to train the neural network and update its weights to improve the ability of failures' prediction. To reach the desired goal, the most two popular aforementioned datasets have been used for NN training and testing. As NN requires mathematical function to adjust the connections' weights, we exploit the tangent hyperbolic function. The details of the algorithm and its setting parameters have been explained comprehensively in [19].

4. RESULT AND DISCUSSION

To assess the proposed MLP-ICA software reliability prediction, testing and validation process have been performed to calculate the evaluation metrics. MSE (Mean Square Error) and Corrected Classification are two important metrics widely used in software reliability prediction techniques.

4.1. ICA-MLP Experimental Results

MSE is a function which identifies the performance of the NN structure. The main goal of ICA algorithm is to minimize the value of MSE not only to overcome convergence problem, but also, to provide more accurate prediction model. MSE can be calculated based on the Formula 1.

$$f(x) = MSE = \frac{1}{n}\sum_{i=1}^{n}(y_{target} - y_{net})^2 \qquad (1)$$

In this formula, y_{target} is the desired output and y_{net} is the output of the network. MSE is computed for both training and testing process in this study. MSE for training samples determines how the network can classify the input data, while MSE for testing samples specifies how the network can memorize or predict the correct output based on the training patterns. On the other hand, Corrected Classification specifies with which probability the network classifies the training and testing patterns correctly. The numerical results of proposed model, performed on two datasets, have been presented in Table 2.

Table 2: Results of ICA-MLP for Pham& Pham and DACS dataset

Parameter	Value	Parameter
MSEtrain (Best Cost)	0.0133	0.0020
MSEtest	0.0078	0.0516
Correct Classification Train (%)	0.9824	0.9987
Correct Classification Test (%)	0.9914	0.9988
Precision (%)	0.631579	0.75
Recall (%)	0.571429	0.6
F-Measure (%)	0.6	0.6666667

The results' comparison showed that there is a direct relationship between the size of dataset and system accuracy. The results indicate that increasing the size of dataset and consequently the numbers of training samples improves the accuracy of the system through the minimizing of MSE. It happens due to this fact that a high number of training data samples leads effectively training the NN structure. Thus, in this condition, the tolerance of network increases steadily and can remember or classify testing data samples with high accuracy.

Having less number of samples (as it is illustrated in Table 2 for Pham & Pham) causes that the correction classification and accuracy of the network to be decreased and consequently MSE will be increased. Correction classification (CC) value of training samples identifies the network's capability in order to classify the training samples. In addition, in testing stage, CC presents the network's capability to remember the training data samples and, according to this, classify the testing data samples. The outputs of NN for both training and testing data samples for DACS dataset are illustrated in the Figures 3 and 4, respectively.

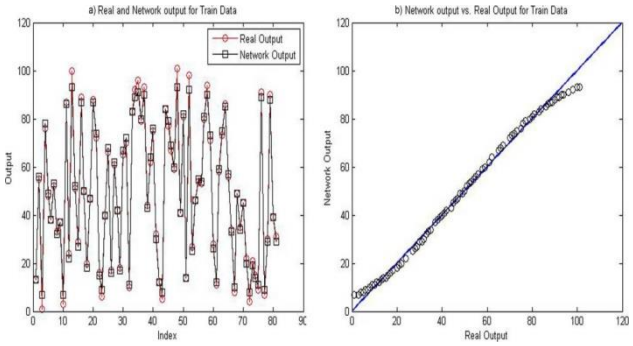

Figure 3 a) Real and network output, b) Desired and network outputs' deviation of the benchmark line (DACS dataset as training samples).

As illustrated in Figure 3, the prediction accuracy is higher as the circles are closer to the line. In Figure 3 (a), the desired and network output similarity is very high due to the proper size of training samples and adequate number of decades parameter identified in the algorithm. Figure 3 (b) proves that the results is near to the benchmark, so the accuracy of the model is acceptable. Similarly, in Figure 4, the output for testing samples for DACS dataset is evaluated.

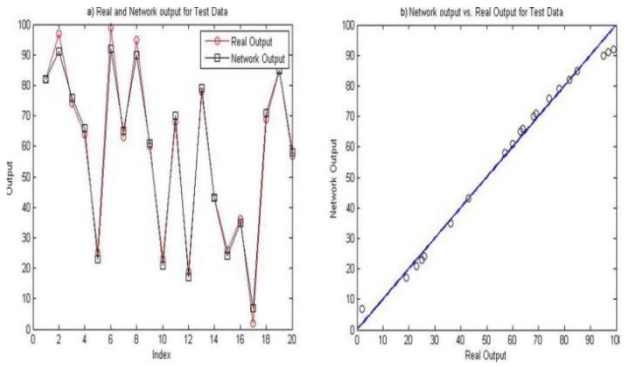

Figure 4 a) Real and network output, b) Desired and network outputs' deviation of the benchmark line (DACS dataset as testing samples).

In Figure 4 (a), in comparison with training result the accuracy is reduced, while the similarity is still high. In NN the most vital issue is dividing training and testing data samples effectively to improve the model prediction in testing phase. Figure 4 (b) illustrates that the system can memorize the training samples with high accuracy.

In next phase, the similar assessments are performed for desired results and network's outputs for Pham & Pham dataset. The results showed that the proposed model is more accurate for big datasets than the small ones. Figure 5 (a) and (b) present the comparison of desired and real output of NN and shows its deviation of the benchmark line.

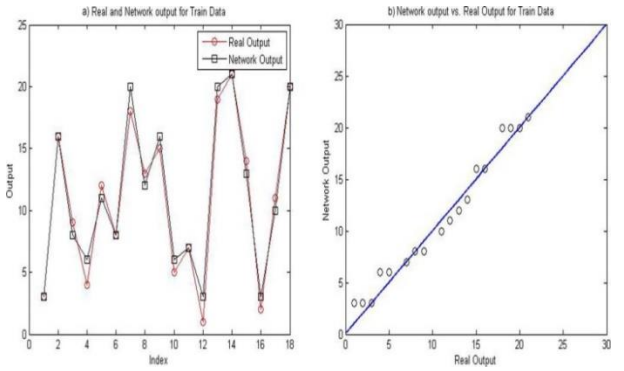

Figure 5 a) Real and network output, b) Desired and network outputs' deviation of the benchmark line (Pham & Pham dataset as training samples).

From these figures, it can be concluded that the model is more efficient for large dataset than the small dataset. In Figure 5 (a) some points illustrate that the network is not accurate for reliability prediction. In addition, in (b) the results are different from desired output. This trend, also, exists in testing samples as shown in Figure 6.

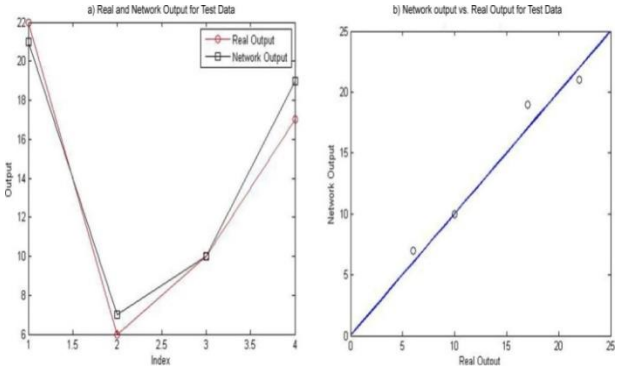

Figure 6 a) Real and network output, b) Desired and network outputs' deviation of the benchmark line (Pham & Pham dataset as testing samples).

4.2. Analytical Models for Reliability Prediction of Software

Statistical models for software reliability prediction exploit complex computation formulas and a large amount of datasets. Moreover, their predictability is very low as they cannot memorize or to be trained for future failures prediction. They only can estimate the current failure status. Logarithmic, Inverse Polynomial, Exponential, Power and Delayed S-shape are the most popular statistical models applied for software reliability prediction.

Logarithmic model never become stable and continues with no bound, so this model cannot be efficient for software reliability which needs convergence phenomena. Exponential model mostly use in growth population methods. Unlike logarithmic model, the trend of exponential model is slow in beginning and then increases rapidly, but their weaknesses are similar. Polynomial model is used to solve non-linear problems by optimizing parameters, but cannot be applied for software reliability prediction due to requiring high polynomial order. Power model is based on the NHPP model. It needs high amount of data and computational works to perform desired task which cause increasing the risk and cost of the project and reducing the accuracy. Finally, Delayed S-shape model improves the exponential model by adding the experiments of the project team's member during growth model process. This model makes wrong assumption which is the software is stable and never changes until the failure is occurred. Goel-Okumoto model and Ohba Model are two types of analytical models which are based on non-homogeneous Poisson process, so they have the same problem as Power model and other NHPP based models including high amount of computational formula and slow speed. They cannot forecast the reliability without having sufficient amount of data.

In the final analysis, the results of comparison among proposed model and analytical models are presented in Table 3.

Table 3: Comparison MSE between analytical models and ICA-MLP model

Model	Training	Testing
Logarithmic	11.61	21.59
Inverse polynomial	7.88	11.97
Exponential	12.85	23.81
Power	14.32	19.65
Delayed S-shape	19.78	30.21
Goel-Okumoto Model	0.080	0.12
Ymada S-shaped Model	1.1467	3.0321
Ohba Model	0.19	0.10
ICA-MLP (proposed	0.0014	0.013

As presented in Table 3, all the weaknesses mentioned earlier caused that the analytical models have the lower accuracy compared with proposed ICA-MLP model. Among analytical models, Goel-Okumoto Model has predicted the reliability with less MSE due to its optimized parameters.

4.3. Evaluation of Neural Network Model for Software Reliability Prediction

In this section, the proposed model will be compared with other NN techniques based on their learning algorithm and network architecture. Most of NN training algorithms follow the randomization approach to initialize the weights of connections of network. Accordingly, different weights' convergence at the end of each training phase will be produces and the prediction results maybe different but very close to each other. Training should be repeated and the average result of all outputs should be calculated. In section 4.2, analytical models were evaluated and it has been found that due to limitation of analytical models, soft computing techniques especially NN approaches have become more popular. There are some benefits of using NNs and more specially MLP, which is a type of feed forward NNs. In Table 4, some features of applying NN for software reliability prediction have been compared with analytical models.

The only disadvantage of using NN is variety results during training process. But also with this weakness the results and accuracy is better than any other analytical approaches.

Table 4: Comparison between Analytical models and Neural Network models

	Neural Network	Analytical
Design	11.61	21.59
Inverse polynomial	7.88	11.97
Function Scope	12.85	23.81
Adaptability	14.32	19.65
Assumption	19.78	30.21
Performance and Efficiency	0.080	0.12

In this section, the proposed model is compared with other NN models respect to the training algorithms and NN architectures.

To achieve the best software reliability prediction model, researchers proposed many different types of NN architectures. Among these models four architectures have become more popular including Multilayer Perceptron, Redial Basis Function NN, Elman Recurrent NN and Fuzzy NN. In this part the comparison between proposed model and these models for software reliability prediction is presented in Table 5.

Table 5: Comparison MSE among variety Neural Network architectures

Model	Training	Testing
ICA-MLP	0.0020	0.0516
RBFN	1.6465	0.1591
Elman	0.1625	0.1394
ANFIS	1.3364	0.9079

The comparison in Table 5 proved that the proposed model gives the best result for both training and testing dataset as it benefits of the proper learning algorithm to adjust the weights and to increase the convergence speed. In the second position, Elman network give the best result as this architecture considers the dynamic behavior of the system and software reliability datasets. RBFN assigns the equal priority (importance) to all input samples which reduces the convergence rate. Moreover, it takes longer time to run which leads to increase the cost and risk of the projects.

ANFIS has given the worse results not only for training, but also for testing due to complexity in its architecture since it makes the combination of two heavy structures including NN and Fuzzy Logic. This composition method maybe useful for some other domains, but for software reliability prediction, the results show that it is not suitable. In conclusion the results have proved that the ICA-MLP gives better results compared with other architectures.

The proposed model also is compared with other NN architectures. The experimental results based on the NRMSE are presented in Table 6 and Figure 7. NRMSE for individual methods can be calculated according to Formula 2.

$$NRMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} {y_i}^2}} \qquad (2)$$

Table 6: Comparison NRMSE among Individual Techniques for Neural Network training algorithm and ICA-MLP

Model	NRMSE
BPNN	0.145541
TANN	0.150355
PSN	0.157922
MARS	0.15267
GRNN	0.166883
MLR	0.147881
TreeNet	0.161121
DENFIS	0.147641
ICA-MLP	0.08574

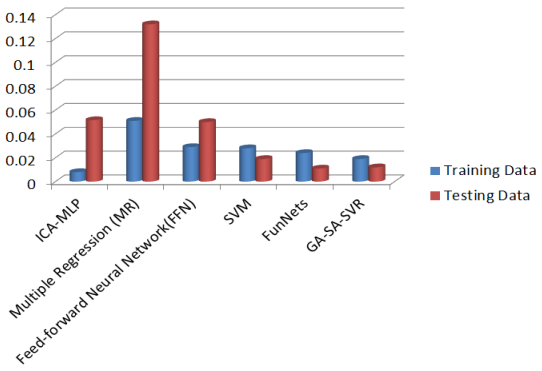

Figure 7 Comparison between ICA-MLP and Soft Computing models

Nowadays, researchers try to propose the models which are based on the combination of different techniques to impart all their advantages and overcome their weaknesses. For software reliability prediction, developers have proposed some ensemble models. In these models, the output of each part is calculated and then they are given into the desired methods as the input. The experimental results based on the NRMSE are presented in Table 7.

Table 7: Comparison NRMSE among Ensemble Techniques for Neural Network training and ICA-MLP

Model	NRMSE
Linear Ensemble (Average)	0.143424
Linear Ensemble (Mean)	0.143463
Linear Ensemble (Median)	0.143399
ICA-MLP	0.08574

In conclusion, the experimental results proved that the ensemble methods are more accurate than individual ones, but the difference is not noticeable as it increases the cost and risk, and need many computational phases. Obviously, in complex projects, these methods cannot be used, since they are difficult to be implemented.

Randomization technique is used for NN training to initialize the connections' weights of network. For each training phase, various weights' convergence is produced. Despite of the fact that the prediction results are different due to different initialization of weights, they are so close to each other. To overcome this inconsistency, the training procedure is run for few times, the results will be taken and, finally, the outputs' average is computed. The investigations have proved that soft computing techniques such as NN give more accurate prediction than analytical methods which are no longer effective.

In NN analysis, this fact should be considered that output results will be different during each training procedure. As the training methods randomly initialize NN weights, even with same training algorithm, network architecture and dataset, still the results will be different. Despite of this weak point, their performance and accuracy are more significant than other analytical techniques.

5. CONCLUSION

In this study, the combination of MLP NN with ICA algorithm is proposed. This model is exploited to predict the future number of software failures to increase the system reliability. In addition, the effectiveness and performance of hybridized model for reliability prediction were analyzed. The results showed that ICA-MLP model is effective for both small and large datasets, while existing models are usually suitable for one type of dataset. The model has less complexity compared with others, especially in mathematical formulas, as the evolutionary pattern has been applied. The results also proved that the model has the lowest MSE and NRMSE in comparison with existing analytical and soft computing techniques.

As in this study only NN technique has been exploited for software reliability prediction, it has been suggested due to uncertainty nature of software, the neuro-fuzzy approach also applied to improve the accuracy.

6. REFERENCE

[1] http://webstore.ansi.org/RecordDetail.aspx?sku=R-013-1992

[2] N. Karunanithi, D. Whitley and YK. Malaiya, Prediction of software reliability using connectionist models, IEEE Transactions on Software Engineering, 1992.

[3] WA. Adnan and MH. Yaacob, An integrated neural-fuzzy system of software reliability prediction, In Software Testing, Reliability and Quality Assurance, Conference Proceedings., First International Conference, pp. 154-158, 1994.

[4] JY. Park, SU. Lee and JH. Park, Neural network modeling for software reliability prediction from failure time data, Journal of Electrical Engineering and Information Science. 1999.

[5] N. Karunanithi, D. Whitley, YK. Malaiya, Using neural networks in reliability prediction, IEEE Software, 1992.

[6] KY. Cai, L. Cai, WD. Wang, ZY. Yu and D. Zhang, On the neural network approach in software reliability modeling, Journal of Systems and Software, 2001.

[7] L. Tian and A. Noore, On-line prediction of software reliability using an evolutionary connectionist model, Journal of Systems and Software, 2005.

[8] L. Tian and A. Noore, Evolutionary neural network modeling for software cumulative failure time prediction, Reliability Engineering & system safety, 2005.

[9] Z. Jun, Prediction of software reliability using connectionist models. Expert Systems with Applications, 2009.

[10] C. Jin, Software reliability prediction based on support vector regression using a hybrid genetic algorithm and simulated annealing algorithm, IET software, 2011.

[11] C. Jin and SW. Jin, Software reliability prediction model based on support vector regression with improved estimation of distribution algorithms, Applied Soft Computing, 2014.

[12] J. Park and J. Baik. Improving software reliability prediction through multi-criteria based dynamic model selection and combination, Journal of Systems and Software, 2015.

[13] Y. Wu and R. Yang. Study of software reliability prediction based on GR neural network, In Reliability, Maintainability and Safety (ICRMS), 9th International Conference, pp. 688-693, 2011.

[14] MC. Liu, W. Kuo and T. Sastri, An exploratory study of a neural network approach for reliability data analysis, Quality and Reliability Engineering International, 1995.

[15] PT. Chang, KP. Lin and PF. Pai, Hybrid learning fuzzy neural models in forecasting engine system reliability, In Proceeding of the fifth Asia Pacific industrial engineering and management systems conference, pp. 2361-2366, 2004.

[16] VN. Vapnik and V. Vapnik, Statistical learning theory, New York: Wiley, 1998.

[17] MK. Bhuyan, DP. Mohapatra and S. Sethi, Prediction strategy for software reliability based on recurrent neural network. In Computational Intelligence in Data Mining, pp. 295-303, 2016.

[18] PR. Bal, N. Jena and DP. Mohapatra, Software reliability prediction based on ensemble models, In Proceeding of International Conference on Intelligent Communication, Control and Devices, pp. 895-902, 2017.

[19] S. Noekhah, AA. Hozhabri and HS. Rizi, Software reliability prediction model based on ICA algorithm and MLP neural network, In e-

Commerce in Developing Countries: With Focus on e-Security (ECDC), 7th International Conference, pp. 1-15, 2013.

ACKNOWLEDGMENTS

This work is supported by Ministry of Higher Education (MOHE) and Research Management Centre (RMC) at the Universiti Teknologi Malaysia (UTM) under Research University Grant Category (R.J130000.7828.4F719).