CHAPTER 2

LITERATURE SURVEY

Karunanithi, N. et al. [27] (1991) had shown that feed-forward neural networks can be developed for predicting software reliability growth. Two important characteristics of neural net models are: consistent behavior in prediction, and a performance comparable to that of other parametric models. They do note, however, that these results are based only on a very few data sets and it is necessary to further test these ideas with a large number of data sets. They were investigated several issues: i) enhancement to the predictive capability of their feed-forward networks approach and ii) application of re-current network models instead of feed-forward networks.

Littlewood, B. et al. [28] (1991) stated that by far the greatest effort in modelling software reliability has gone into the 'reliability growth' problem. There are quite stringent requirements for the testing regime in which data is collected, but if these are satisfied it is now possible to obtain accurate reliability estimates, and predictions of future failure behaviour, and to know that these are accurate. Whilst the techniques work well for modest reliability levels, it seems unlikely that they would ever be suitable for those situations, such as certain safety-critical applications, which require ultra-high reliability. These observations place a practical limit on the level of reliability that can be assured by reliability growth techniques. Other approaches to this problem will be discussed briefly and critically. They concluded that there is no scientific basis upon which to conduct the validation of software with ultra-high dependability requirements.

Karunanithi, N. et al. [29] (1992) demonstrated how we can use neural-network models and training regimes for reliability prediction. Results with actual testing
and debugging data suggest that neural-network models are better at endpoint predictions than analytic models. Though the results presented here are for only one data set, the results are consistent with 13 other data sets they tested. They recognize that their experiments are only beginning to tap the potential of neural-network models in reliability, but they believe that this class of models will eventually offer significant benefits. They also recognize that their approach is very new and still needs research to demonstrate its practicality on a broad range of software projects.

Karunanithi, N. et al. [30] (1992) demonstrated the applicability of neural network models with a clipped linear output unit. This modified structure paves way for applying neural networks without prescaling the data. Among the proposed models the Jordan network model exhibits a better accuracy than the feed-forward model. We developed expressions for the network structures and showed how the underlying failure rate process $\lambda(t)$ can be derived. These expressions could be used for further analysis on models developed by the neural network approach. In their research they restricted the applicability of the neural networks to model only the cumulative faults. However this need not be the case. Since the neural network approach is a "black box" approach, models that can incorporate additional knowledge such as the program complexity can be easily developed by adding more input units. This approach is currently being investigated.

Cai, K. Y. et al. [31] (2001) examined the effectiveness of the neural network approach in handling dynamic software reliability data. They used multilayer perceptions to handle time between successive software failures as well as number of software failures as well as number of software failures observed in successive time intervals. Moreover, they examined the effects of network architectures, scaling functions and filtering techniques. While confirming
previous observations to some extent on the neural network approach in software reliability modeling they had several new findings.

Aljahdali, S. H. et al. [32] (2001) had shown that neural network can be used for building software reliability growth models. NNs were able to provide models with small SSE than the regression model in all considered cases. If a regression model with higher order have been considered probably less SSE is obtained. However, the number of the regression model parameters will be increased. This will require more observations for providing reliable estimate of the parameters. At present, we are investigating the use of evolutionary computations in to solve the software reliability growth modeling problem.

Yu-Shen Su et al. [33] (2005) proposed an artificial neural network-based approach for software reliability estimation and modeling. They first explained the network networks from the mathematical viewpoints of software reliability modeling. That is, they would show how to apply neural network to predict software reliability by designing different elements of neural networks. Furthermore, they would use the neural network approach to build a dynamic weighted combinational model. The applicability of proposed model is demonstrated through four real software failure data sets. From experimental results, they could see that the proposed model significantly outperforms the traditional software reliability models.

Sultan H. Aljahdali and Khalid A. Buragga et al. [34] (2008) demonstrated the results of utilizing the connectionist modeling and learning algorithms for prediction the software reliability. The demonstration includes a comparison of the four connectionist models with different leaning algorithms or structures. The Elman recurrent neural networks is a robust technique for function prediction due capturing the dynamic behavior of the data set. The preliminary computational
results in the MATLAB environment seem quite promising and give insight into the generalization capability of these models.

**Raj Kiran et al. [35] (2008)** ensemble models are developed to forecast software reliability efficiently. Three linear ensembles and one non-linear ensemble are developed and tested to forecast software reliability. Various statistical and intelligent techniques constitute the ensembles. They are multiple linear regression (MLR) and multivariate adaptive regression splines (MARS); backpropagation trained neural network (BPNN), dynamic evolving neuro–fuzzy inference system (DENFIS) and TreeNet. Based on the numerical experiments conducted by us on the software reliability data obtained from literature, we noticed that the non-linear ensemble outperformed all the other ensembles and also the constituent statistical and intelligent techniques. Further, they noticed that the linear ensembles also outperformed the constituent techniques from lag3 onwards. In conclusion, the ensembles developed here can be used as viable alternatives to the existing methods for software reliability prediction.

**Nirvikar Katiyar and Raghuraj Singh et al. [36] (2011)** stated that software crisis focuses the attention of software engineers on the research of systematic techniques for software developer in an attempt to make software systems more reliable and qualitative. This calls for more research into building better models. The objective of this study was not to present a definitive model for the prediction of program quality measures, but rather to explore and evaluate neural and regression modeling techniques. Regression models are widely used by software reliability engineers to predict the number of faults in program modules. However, a neural network approach opens a broad new spectrum in our search to obtain models possessing superior quality of fit and prediction. Results from ARE values recorded in the tables suggest that neural network approach possess good properties from the standpoint of model quality of fit and predictive capability. In
order to build a linear regression model it is necessary to first select from a set of independent variables a subset of these variables that will explain the largest amount of variance in a dependent variable. The neural network approach presented here allows the user to directly use the collection of independent variables as the input to the neural net. During the training, the network itself will select the most significant software complexity metric by computing the appropriate weight matrix. As a result the user is relieved from having to perform costly preprocessing, and the resultant model is likely to be superior in performance quality. The regression analysis technique also involves assumptions regarding independence of response (dependent) variables, i.e., number of program faults etc. The neural net approach does not require any assumptions of this nature and allows response variables which are linearly dependent. The subject of predictive quality and appropriateness of the models has failed to receive the attention that it deserves in many reliability models they have studied. In general, there is much performance which aid in the determination of the predictive quality of models. The researcher has examined some in this research. Within the framework of this limited data set, their findings suggest an empirical basis for use of neural network model in order to identify fault-prone program modules.

Wu, Y. et al. [37] (2011) involved an encoding scheme with multiple evaluation metrics and uses neural network to perform clustering algorithm, thus offering new insights into model selection. This new approach is characterized by high precision, simple structure, and fast calculating speed. However, this approach is not without flaws. First of all, 17 failure data sets are used in this research and baseline data have involved in supervised learning for nearly 80 times. Although the accuracy has somewhat been verified in simulation experiment, more failure data should be used and more learning be carried out to further enhance the
accuracy of the network. Next, BP network learning is an important step, where learning sample is essential to model stability and accuracy. The essence of this approach is to automatically optimize standard weight coefficients through sample learning by BP network. Therefore, in order to strengthen the accuracy and stability of the network clustering, further study is needed on how to collect and select empirical data to supervise network learning. Finally, as model selection cannot be separated from model evaluation metrics, research on and advancement of more effective model evaluation metrics will further improve model selection approaches.

Li, Q. et al. [38] (2011) stated that software reliability evaluation plays a very important role in the development of software, but the traditional software evaluation method mostly focuses on evaluation by use of failure data which is gained only after testing or usage in the late phase of the software life cycle. Thus people hope to get every stage’s information about the software’s reliability which is taken as the reference or accordance to guide the software’s design, analysis and testing and so on. A software reliability evaluation method is put forward in this research, which focuses on lots of information correlative with reliability during the whole software life cycle. Finally an application is put forward to demonstrate the feasibility of this method.

Sandhu, P. S. et al. [39] (2012) investigated whether qualitative and quantitative factors can be used to identify level of number of faulty software modules. They compare the performance of Batch Gradient Descent (BGD), Batch Gradient Descent with momentum (BGDWM), Variable Learning Rate (VLR), Variable Learning Rate training with momentum (VLRM) and Resilient Backpropagation (RB) based Neural Network for the fault dataset. Variable Learning Rate training with momentum algorithm shows best results among the five algorithms experimented with least values of MAE and RMSE calculated as 0.5445 and
0.7248 respectively. The Accuracy% of prediction of the level of number of faults for Variable Learning Rate training with momentum algorithm is also highest i.e. 77.4194. The performance of Batch Gradient Descent without momentum algorithm, Batch Gradient Descent with momentum and Resilient Backpropagation algorithms comes out to be the same for the fault dataset used and the results of the Variable Learning Rate without momentum are second best but much below than the best one. It is therefore, concluded the Variable Learning Rate with momentum based neural network model is implemented and the best algorithm for classification of the software components into different level of number of faults present in the modules of the software systems.

Yajnaseni Dash et al. [40] (2012) presented a study predicted the maintainability of object oriented software by using Multi Layer Perceptron neural network model. This MLP model is found to be more appropriate for the estimation purpose. Thus Artificial Neural Networks have shown their ability to provide an adequate model for predicting maintenance effort. The comprehensive experimental analysis showed that the proposed model is quite valuable for real life applications. They would further explore this research field to minimize the issues associated with software maintainability and object-oriented software system.

Amandeep Kaur et al. [41] (2012) said that a variety of software fault prediction techniques have been proposed, which have been already applied in software engineering applications. A neural network is trained to reproduce a given set of correct classification examples, instead to produce formula or rules. Neural networks are non linear techniques that are able to model complex functions. In this work NN approach is used along with PSO to find out fault prone components of software. The evaluation measures used are Accuracy, MAE, RMSE and the results are calculated on different iterations. Result is calculated on
different iterations. The accuracy is improved as the number of iterations increases. The results showed that proposed technique is better as compared to traditional approach and give better results.

**V.Ramakrishna et al. [42] (2012)** developed two network models, one is an MLP with back-propagation algorithm and the other is a Support Vector regression model to predict the reliability of software using the available information. The outcomes of both the models are consistent with the knowledge of the domain expert. Thus in this proposed research, the results support the reliability of the software by both of the networks and also gives an importance of understanding the impact of neural networks for finding the reliability of software.

**Bisi, M. et al. [43] (2012)** proposed a Feed Forward neural network with two encoding scheme such as exponential and logarithmic function. The effect of encoding and the effect of different encoding parameter on the prediction accuracy have been investigated. The effect of number of hidden nodes on prediction accuracy has also been shown. The experimental results show that the proposed approach gives acceptable results for different data sets using the same neural network architecture.

**Chawla, G., & Thakur, S. K et al. [44] (2013)** studied the software reliability is been defined under the software fault analysis. The research has also defined a fault based software cost estimation concept. The fault defined the software criticality under the fault study and based on it software reliability is been defined in this work.

**Aggarwal, G. et al. [45] (2013)** Present the concept of the software reliability based on neural network. The reliability can be optimized based on neural network.
Butey, P. K. et al. [46] (2013) proposed an approach used neural network modelling for the prediction of reliability in any given software. It has been shown that the neural network modelling is applied to represent the existing software growth models in detail. Further the neural-network-based modelling is used to achieve a dynamic weighted combinational model. To validate proposed models, they used a numerical example of real software failure data set. They had compared the performances of the neural network models with some conventional software growth models from point of view of goodness of fit and prediction ability for the faults. The relative error in the prediction is also calculated on the basis of root mean square error. It has been confirmed from the experimental results that the neural network models are workable and have more accuracy with both goodness of fit and the prediction ability of faults compared to existing conventional models.

Adebiyi, A. et al. [47] (2013) integrate the security to the SDLC. The security is integrated at the design phase and the result shows the effectiveness of the techniques.