Neural Network Approach to Measure Reliability of Software Modules: A Review

Gaurav Aggarwal¹, Dr. V.K Gupta²
¹Research Scholar, Jagannath University
⁴⁰auravaggarw@gmail.com
²Assistant Professor, University of Rajasthan Jaipur
vinodgupta602@gmail.com

Abstract: Software reliability is a key part in software quality. It is the ability of the software to perform its specified function under some specific condition. Reliability can be associated with both hardware and software. The hardware reliability can easily be evaluated since hardware get wear out but in case of software it be very difficult. So neural networks have been used for the last few decades in a broad variety of applications. In this paper we have studied the software reliability of neural network based and made a review.

Keyword: Software Reliability, Reliability Model, Neural Network.

Introduction:-

Reliability in the general engineering sense, is the probability. It gives component or system in a define environment will operate correctly for a specified period of time. An important issue in developing such software systems is to produce high quality software system that satisfies user requirements. Software reliability models specify some reasonable form for this distribution, and are fitted to data from a software project. Once a model demonstrates a good fit to the available data, it can be used to determine the current reliability of the software, and predict the reliability of the software at future times.

Software reliability is often defined as —the probability of failure-free operation in a defined environment for a specified period of time. A failure is the departure of software behavior from user requirements. This dynamic phenomenon has to be distinguished from the static fault (or bug) in the software code, which causes the failure occurrence as soon as it is activated during program execution.

Since software does not deprecate like hardware, the reliability of software stays constant over time if no changes are made to the code or to the environmental conditions including the user behavior. However, if each time after a failure has been experienced the underlying fault is detected and perfectly fixed, then the reliability of software will increase with time.

Various approaches can be used to improve the reliability of software, however, it is hard to balance development time and budget with software reliability. The main problem is that the software systems are complex so that software engineers are not currently able to test software well enough to insure its correct operation. Due to the assumptions made by various software reliability models, or due to there is dependence among successive software runs [22, 23]. These problems are:

i) By finding mechanisms or relationships to more accurately determine the quality of software systems, without visiting a large fraction of their possible states.
ii) Taking in consideration the failure correlation and;
iii) Considering there is no single model sufficiently trustworthy in most or all applications.

There are many ways of using parametric models, nonlinear time series analysis and data mining to model software reliability and quality have been investigated. Fuzzy logic, neural networks, genetic algorithm, genetic programming and evolutionary computation are the most important key methodologies. Failure and fault are two different factors which are generally inbuilt in our software during the development phase. Fault can be said as an error or bug which is introduced during the development phase. These investigations point the way towards using computational intelligence technologies to support human developers in creating software systems by exploiting the different forms of uncertainty present in a software system results from infrequent and unpredictable occurrence of human errors and incomplete or imprecise data, in order to model complex systems and support decision making in uncertain environments [24].

Software Reliability is an important attribute of software quality, functionality, usability, performance, serviceability, capability, install ability, maintainability, and documentation. It is hard to achieve, because the complexity of software tends to be high. There are three major classes of software reliability:

1) Black box reliability analysis: Estimation of the software reliability based on failure observations.
from testing or operation. These approaches are called black box approaches because internal details of the software are not considered.

2) **Software metric based reliability analysis:** Reliability evaluation based on the static analysis of the software (e.g., lines of code, number of statements, complexity) or its development process and conditions (e.g., developer experience, applied testing methods).

3) **Architecture-based reliability analysis:** Evaluation of the software system reliability from software component reliabilities and the system architecture (the way the system is composed out of the components). These approaches are sometimes called component-based reliability estimation (CBRE), or grey or white box approaches.

The characteristic of the software reliability are correctness, consistency and precision, robustness, simplicity and traceability. In general, there are two major types of software reliability models: the deterministic and the probabilistic. The deterministic is employed to study the number of distinct operators and operands in the program. The probabilistic represents the failure occurrences and the fault removals as probabilistic events. The probabilistic models can be further classified into different classes, such as error seeding, failure rate, and non-homogeneous Poisson process (NHPP). Among these classes, the NHPP models are the most popular ones. The reason is the NHPP model has ability to describe the software failure phenomenon. The first NHPP model, which strongly influences the development of many other models, was proposed by Goel and Okumoto [25]. Later, Ohba [26] presented a NHPP model with S-shaped mean value function. Yamada and Osaki [27, 28] also made further progress in various S-Shaped NHPP models. Although these NHPP models are widely used, they impose certain restrictions and assumptions about the nature of software faults and the stochastic behavior of software failure process. To solve this problem, several alternative solutions are introduced. One possible solution is to employ the neural network, since it can build a model adaptively from the given data set of failure processes. Many researchers [29-39] have been successfully adapted neural networks to software reliability issues. Many software reliability models have been developed from past three decades. They are developed through either an analytical or data-driven approach. Analytical software reliability growth models (SRGMs) represented by Non-Homogeneous Poisson Process (NHPP), are stochastic models focusing on software failure process. Data-driven models are developed from historical software fault-related data, following the approach of regression or time series analysis.

The survey been conducted in this paper is summarized in different section. In first section of the introduced basic the concept of software reliability. In the second section of the paper will introduced the role of neural network in software reliability model till been designed and the criteria been used. In the third section of the paper will discuss the neural network architecture, basic properties and learning technique.

**Role of Neural Network In Software Reliability:**

The reliability model till now been designed are based on the study of failure associated with the code and the environment where it is been implemented [20]. All the software reliability models are designed on the basis of execution time and calendar time. Execution time of any program is the time that is actually required or spent by the processor in executing the instruction of that program [21]. Calendar time is referred as the elapsed time from start to end of program execution on a running computer.

Neural network is a collection of fast processing and computing nodes called artificial neurons. These neurons are designed on the basis of study of the behavior of biological neuron. These neurons are connected in a specific manner that is layer like structure known as neural network architecture. Neural network is an output based computing technique. It is a technology generally been used for optimizing problem. Neural networks also consist of multiple layers of computational units, usually interconnected in a feed-forward way. Neural network [41] has been applied to estimate parameters of the formal model and to learn the process itself in order to predict the future outcomes. It has been shown that feed forward network can be applied for prediction. Back-error propagation is one of the most widely used neural network paradigms and has been applied successfully in application studies in a broad range of areas [40].

Each neuron in one layer has directed connections to the neurons of the subsequent layer. In many applications the units of these networks apply a sigmoid function as an activation function. ANN (Artificial Neural Network) software reliability models have recently aroused more research interest [6, 7, 8, 9, 14, 15, 17]. Traditionally, both kinds of models only consider single fault detection process (FDP) and data for analysis are only from FDP. However, while data from both FDP&FCP (fault correction process) are available, NHPP and ANN models can be extended into paired NHPP models and combined ANN models, providing more accurate predictions [3,6]. Generally speaking, data-driven approach is much less restrictive in assumptions compared to analytical approach.

**Architecture Of Neural Network:**

The artificial neural network is designed on the basis of the study of biological neural network. A human brain is
responsible for all the action and reaction taken by human body. These actions are controlled by brain and are carried
to different parts of the body by a fast processing node
known as neurons. Similarly in case of artificial neurons
also we have a fast processing node responsible for
performing the computation and are known as Neurons [19]. A neural network is a collection one to multiple
neurons which are arranged in a specific manner to perform
the computation. The basic structure of a neuron is
represented in the given below figure:

The model is designed for a set of input values and
and corresponding desired output. Suppose consider that „I”
is the set of input values such that I={i1,i2,……..in} and we
associate a variable parameter with it which controls the
behavior of neurons. Let us consider weights associated
with the input as the variable parameter such that
W={w1,w2,…….wn}. All the input along with the weight
is passed through the summation unit where compute the
net input which given by

$$\text{Net}_{\text{input}} = \sum_{i=1}^{n} iwi \text{ where } n = \text{number of input to the network } \ldots \ldots$$

= i1 *w1+i2*w2+i3*w3+.......... +i|wi

These neurons are arranged in different layers to represent a
specific structure known as neural network architecture. For
designing a reliability model we will require a multilayer
of neurons. The neurons are arranged in three different layers
known as input layer, hidden layer and output layer. The
multi layer network is shown in below Figure2:

**Comparison of Neural Network technique with other
techniques:**

Here various techniques which are used for the predictions
of software reliability are enlisted.
The following techniques are applied to predict software
reliability:-

**Threshold-acceptance-based neural network [42]:**
Threshold accepting (TA) is a faster variant of the original
Simulated Annealing algorithm wherein the acceptance of a
new move or solution is determined by a deterministic
criterion rather than a probabilistic one. The predominant
aspect of TA is that it accepts any new solution, which is
not much worse than the current one. The crux of the TA-
based training algorithm for the feed forward neural
networks is that the ‘forward pass’ of the back propagation
algorithm is not disturbed and retained ‘as it is’. But, in the
backward pass, which essentially updates all the weights,
TA is used instead of the steepest descent algorithm used in
back propagation. In this context, the set of weights of the
neural network (both input to hidden and hidden to
output nodes) becomes the vector of decision variables. The
second author coded TANN.

**Pi–Sigma network (PSN) [43]:** The Pi–Sigma network
(PSN) is a feed-forward network with a single hidden layer,
where the number of hidden units (also called as ‘summing
units’) represents the order of the network, which can be
varied as required. In the output layer there are product
units whose output is a function of the product of the
individual summing units’ output. In every iteration of the
algorithm, until the convergence criterion is met, one of the
summing units will be selected at random and the
 corresponding weights of the links connected to that node
are updated according to a rule similar to the delta rule.

**Multivariate adaptive regression splines (MARS) [44]:**
MARS is an innovative and flexible modeling tool that
automates the building of accurate predictive models for
continuous and binary dependent variables. It excels at
finding optimal variable transformations and interactions, the complex data structure that often hides in high-dimensional data. In doing so, this new approach to regression modeling effectively uncovers important data patterns and relationships that are difficult, if not impossible, for other methods to reveal.

**Generalized regression neural network (GRNN) [45]:**
GRNN is a normalized radial basis function (RBF) network in which there is a hidden unit centered at every training case. These RBF units are called “kernels” and are usually probability density functions such as the Gaussian. The hidden to output weights are just the target values, so the output is simply a weighted average of the target values of training cases close to the given input case. The only weights that need to be learned are the widths of the RBF units. These widths (often a single width is used) are called “smoothing parameters” or “bandwidths” and are usually chosen by cross-validation or by more esoteric methods that are not well known in the neural net literature; gradient descent is not used. GRNN is a universal approximator for smooth functions, so it should be able to solve any smooth function approximation problem given enough data. The main drawback of GRNN is that, like kernel methods in general, it suffers badly from the curse of dimensionality. GRNN cannot ignore irrelevant inputs without major modifications to the basic algorithm.

**TreeNet [46]:** TreeNet makes use of a new concept of “ultra slow learning” in which layers of information are gradually peeled off to reveal structure in data. TreeNet models are typically composed of hundreds of small trees, each of which contributes just a tiny adjustment to the overall model. TreeNet is insensitive to data errors and needs no time-consuming data preprocessing or imputation of missing values. TreeNet is resistant to overtraining and is faster than a neural net.

**Dynamic evolving neuro-fuzzy inference system (DENFIS) [47]:**
DENFIS evolve through incremental, hybrid (supervised/unsupervised) learning, and accommodate new input data, including new features, new classes, etc., through local element tuning. New fuzzy rules are created and updated during the operation of the system. At each time moment, the output of DENFIS is calculated through a fuzzy inference system based on most activated fuzzy rules, which are dynamically chosen from a fuzzy rule set. A set of fuzzy rules can be inserted into DENFIS before or during its learning process. Fuzzy rules can also be extracted during or after the learning process.

While comparison is made among these techniques separately it is observed that all the techniques differ in their working criteria and have some performance problem. Thus, [48] Ensemble models are developed to accurately forecast software reliability. The idea behind ensemble systems is to exploit each constituent model’s unique features to capture different patterns that exist in the dataset. Both theoretical and empirical works indicate that ensemble can be an effective and efficient way to improve accuracies.

Some of the ensemble techniques for prediction problems with continuous dependent variable include linear ensemble (e.g., simple average; [49]), weighted average [50], stacked regression [51] and non-linear ensemble (e.g., neural-network-based nonlinear ensemble [52]). An ensemble uses the output obtained from the individual constituents as inputs to it and the data is processed according to the design of the arbitrator. Four different variants of ensembles are designed and employed as shown in Figures. 1 and 2. These ensembles are described briefly below:

1. **Linear ensemble based on average:** For each observation, the output values of the individual components are taken as the input to the ensemble and the average of these values is output by the ensemble. This is the simplest kind of ensemble one can imagine.

2. **Linear ensemble based on weighted mean:** In this ensemble, the individual output values are not taken as they are but are given weights based upon certain criteria set by the user. In this case, the criteria of setting the weightages is based on the mean of the normalized root mean square error (NRMSE) values over the individual lags on the test data. The lower the mean the higher the weightage with the condition that the sum of all the weights is equal to one. This helps in setting the priority towards a technique based on its performance.

3. **Linear ensemble based on weighted median:** It is similar to the linear ensemble based on weighted mean, except that the median of the NRMSE values of the individual techniques on the test data is considered in assigning the weightages instead of the mean of the values.

4. **Neural network based non-linear ensemble:** Here, no assumptions are made about the input that is given to the ensemble. The output values of the individual techniques are fed into an arbitrator, which is a back propagation neural network (BPNN) which when trained, assigns the weights accordingly.
We noticed that the non-linear ensemble outperformed all the other ensembles and also the constituent statistical and intelligent techniques because in non-linear ensemble neural-network is used due to which performance is improved.

**Related Work**

Karunanithi et al. [11,13,16,18] first applied neural network architecture to estimates the software reliability. They also illustrated the usefulness of connectionist models for software reliability growth predictions. Cai et al. [10] examined the effectiveness of the neural network approach in handling dynamic software reliability data overall and present several new findings. They found that the neural network approach is more appropriate for handling datasets with ‘smooth’ trends than for handling datasets with large fluctuations and the training results are much better than the prediction results in general. Sherer [12] has applied neural networks for predicting software faults in several NASA projects. Khoshgoftar and Szabo [4] used the neural network as a tool for predicting the number of faults in a program and concluded that the neural networks produce models with better quality of fit and predictive quality. Su and Huang proposed [5] Dynamic Weighted Combinational Model (DWCM) that follow neural network based approach to software reliability. Kapur, Khatri and Goswami[7] have proposed a Generalized Dynamic Integrated Model (GDIM) using ANN approach, which incorporates the concept of $n$ types of faults. Kanmani et al. [1] proposed two neural network based fault prediction models using object-oriented metrics. R. C. Tripathi, Manohar Lal, proposed [2] the software reliability of systems with the help of past fault-related data sets by using Artificial Neural Networks (ANN) models. Numerical examples are shown with both actual and simulated datasets. Better performance of software reliability assessment is observed, compared with original ANN model with no such historical fault-related data incorporated.

**Conclusion:**

The main purpose of this paper was to understand the concept of software reliability using the artificial neural network. This paper introduces the concept of neural model and its architecture. In future we are going to design a neural model for calculating the reliability. A neural network is considered as an optimizing technique which is used to scale the output.

**References:**


[21] By KK Agarwal and Yogesh Singh “Software Engineering”.


