Advances and Applications in Mathematical Sciences
Volume 20, Issue 12, October 2021, Pages 3007-3015

CONVOLUTIONAL NETWORKS FOR OPTIMAL SELECTION OF SOFTWARE RELIABILITY ASSESSMENT GROWTH MODEL

CHANDRA MOULI VENKATA SRINIVAS AKANA, CH. DIVAKAR and CH. SATYANARAYANA

Department of Computer Science
AMC Engineering College
Bengaluru, Karnataka, India
E-mail: mouliac@yahoo.co.in

Department of CSE
SRKR Engineering College
Bhimavaram, Andhra Pradesh, India
E-mail: divakar_c@yahoo.com

Department of CSE
JNTUK
Kakinada, Andhra Pradesh, India
E-mail: chsatyanarayana@yahoo.com

Abstract

Abstract- In literature the "Software-Reliability" of a product is assessed by software growth models (SRGMs). In research, software reliability is considered as the essential parameter by which the quality of the software can be measured. So in present scenario, most of the software development and service oriented companies are bound and trying to provide high reliable software to gain confidence and sustainability in the market. Multiple Software growth models are present in literatures however most of them does not works across different environments. Selection of optimal model is of utmost importance, in this paper the concept of convolutional neural networks is used to predict and select the most optimal growth model for the environment.

2010 Mathematics Subject Classification: 68T07.
Keywords: software reliability growth models; deep learning; reliability assessment.
Received October 6, 2020; Accepted October 26, 2020

I. Introduction

In terms of English the term "Software Reliability", is characterized as the capacity of the software to withstand and works with no failure. In Literature, several software growth models have been proposed with the only intension of measuring the reliability of the software. These models are highly available that often raises a question that, "Is every model is able to fit every dataset?". So to answer this question, an inclusive and detail analysis of software-reliability growth model is required.

The main challenge for every software engineer is to develop high quality software for that; the developed software has to pass through a sequence of tests which declare that how sustainable the software is and its quality. The main limitation of any quality software is occurrence of faults which degrade the quality and makes the software unreliable that may not meet the customer requirements and satisfaction.

Several parameters were employed to find the number of faults with the working software growth model; one such type of parameter is mean time to failure and another parameter is cumulative distribution function. Based on this analysis these SRGMs are classified as Parametric [1] and Non parametric [2, 3] models. Since 1972, the failure occurrences were evaluated during the testing phase and then the reliability assessment could be announced. Many of the traditional SRGMs assessment was relied on mean value function and this function would be Exponential-growth [4], S-Shaped Growth model [5], this may be both [6] [7]. Apart from these, it becomes difficulty for the managers to choose the most suitable software-reliability growth model for project development cycle. For instance, managers assess the reliability with a growth model for the past information and then use the assessment criterion for the model to make a decision of choice. However, the estimation results that were obtained from the past data may not guarantee the future data values which may be differed from actual software project data. In this paper, mathematical analysis of convolutional network is designed for the selection of most suitable reliability growth model. This analysis includes multiple numerical analysis, examples that uses the fault data and some synthetic software fault data.

II. Related Works

In this part a concise survey of literature on reliability assessment of SRGMs were presented.

In [8], a combination of genetic algorithm and bagging algorithm was proposed by Wahono and others. In this work they have employed the genetic algorithm for feature selection and bagging algorithm for solving the problem of class imbalancing. I this work, the authors have applied the model on NASA dataset and compared the outcomes with conventional binary classifiers like SVM, DT and Neural Networks. From the experimental results that were obtained shows that there is a considerable improvement in the prediction and a score of 89.9% was attained with SVM.

In [9] Wang others have ensemble the feature rankings technique and compared it to filter based ranking techniques. This includes some statistical parameters like gain-ratio, Gain and the models were built with logistic regression, SVM and KNN classifiers. For this investigation, the authors have considered three datasets from PROMISE data repositories. The outcomes acquired uncover that the group approach has the improved exhibition when analyzed against singular rankers.

In [10], Li and others have proposed, defect detection with CNN. In this work, the authors have the structural and statistical features of the program and that were trained automatically to the classifier. In this analysis they have considered four phases which are abstract syntax and are utilized to extract the tokens which are encoded into some numerical values. In later stages of the approach these are combined and processed to a CNN where they are combined with traditional defect predictive feature set. At conclusive stage a logical regression was used to choose whether the code files are having bugs or not. In these experiments it was demonstrated that the proposed CNN provided extensive improvement when tried on seven open source projects

A prediction model was developed to learn the features automatically which was proposed by Dam et al. in [11]. This work aims to use the source code for predicting the defects, in this work the authors have used a chain like structure similar to tree branches and employed LSTM (Long Short Term

Memory) network that expects to coordinate the best abstract syntax branch. This function is able to detect the defectives automatically whether it may be within the project or in some other different project.

III. Background

Several Software-Reliability growth models have been developed and applied for the projects to assess the quality of it, thereby providing a quality based software as a part of quality management and testing process control of the project development Cycle. To facilitate this, parameters like mean-value function is employed to detect the faults during an interval of [0 t]. In this section, few SRGMs that are related to the current work were discussed.

(1) Exponential Non Homogeneous Poisson Process Model

$$E(t) = a(1 - e^{-bt}) \tag{1}$$

(2) Delayed S-Shaped Non-Homogeneous Poisson Process Model

$$S(t) = a\{1 - (1 + bt)e^{-bt}\} \tag{2}$$

(3) Logarithmic Poisson Execution Time Model

$$\mu(t) = \frac{1}{\theta} \ln [\lambda_0 \theta_t + 1] \tag{3}$$

(4) Exponential Stochastic Differential Equation Model

$$SDE_g(t) = a\{1 - e^{-bt + \frac{a^2}{2}t}\} \tag{4}$$

(5) S-Shaped Stochastic Differential Model

$$SDE_s(t) = a\{1 - (1 + bt)e^{-bt + \frac{g^2}{2}t}\} \tag{5}$$

In all the above equations from 1 to 5 term 'a' represents the number-of-inherent faults, 'b' represents the fault-detection-rate per unit time λ0 represents the inherent failures, 'θ' is the reduction rate, a σ is defined as a positive constant representing the magnitude with irregular fluctuations.

IV. Convolutional Neural Networks for Optimal Growth Model Selection

To build up an exceptionally reliable Software System; certain factors are to be considered during the testing phase like

(i) c1: Cost per fault

(ii) c2: Cost per unit time

(iii) c3: maintenance cost per fault

So, the Software cost can be mathematically formulated as

$$C_1(t) = c_1 H(t) + c_2 \tag{6}$$

In the above equation the term "$H(t)$" represents the mean value function of the employed SRGM. From the above analysis the total expense for the maintenance of the software can be mathematically represented as

$$C_2(t) = c_3\{a - H(t)\} \tag{7}$$

From the above 6 and 7 equations the total estimated software cost will be given as

$$C(t) = C_1(t) + C_2(t) \tag{8}$$

By minimizing the "t" in $C(t)$ of equation (8), the optimum release time can be given.

The weights in the sensory layer of the network is represented as $w_{ij}^1(I = 1, 2, \ldots, I; j = 1, 2, \ldots, j)$ where i^{th} sensory unit and j^{th} association layer and the connection weights are termed as $w_{ij}^1(j = 1, 2, \ldots, j; k = 1, 2, \ldots, K)$ for j^{th} unit and k^{th} unit of response layer. The input for the sensory layers are x_i and the outputs are y_i when these are applied with detected faults per unit time N_i.

Considering all above limitations for the software reliability model then the accompanying amount of information as parameters $\lambda_i(I = 1, 2, \ldots I)$ to the input data $x_i(i = 1, 2, \ldots, I)$.

- Akaike's Information estimated values
- Mean square error estimation
- Estimation of all the parameters of the model
- Estimation is carried out for 25%, 50% and 75 % of the faults for the considered datasets

So the rules for input-output for each layer is given as

$$h_j = f(\sum_{i=1}^{I} W_{ij}^1 x_i) \tag{9}$$

$$y_k = f(\sum_{j=1}^{J} w_{jk}^2 h_j) \tag{10}$$

Thus, in this work a multi layered back propagation neural network is considered to learn the interaction among the input and output [12].So, an error function is formulated and given as

$$E = \frac{1}{2}\sum_{k=1}^{K}(y_k - d_k)^2 \tag{11}$$

In this work to analyze the output values; different models like exponential NHPP models and others mentioned in section 3 are considered. The number of 'i' sensory units is 37 and 5 models, so in this analysis a total of 5 models with two growth models are taken. The estimation results with 90% of faults and 80% of faults were tested and the analysis is shown in later sections. From the investigation, it very well may be affirmed that the model is best fitted for the past data and isn't generally fitted for future predictions. So the problems arise for the managers to opt the most suitable model that could able to detect the faults accurately

V. Test Analysis

This section presents the estimated results that were obtained with optimal model using traditional network and convolutional network. These analysis are shown in the tables 1 and 2, so from the investigation it is

discovered that estimated recognition rates with convolutional network accomplishes better accuracy than that of conventional neural network. In this analysis, the assessment results depend on convolutional network accomplish more recognition rate even for 80% flaw dataset altogether higher than conventional network.

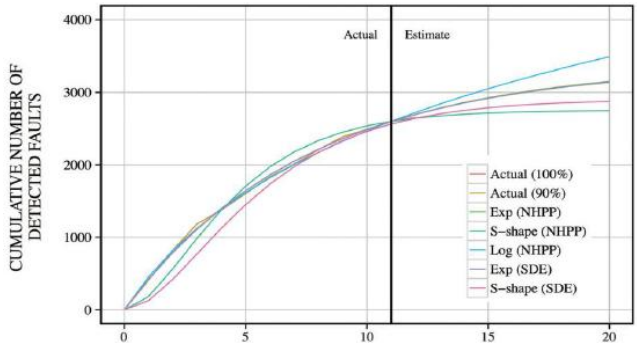

Figure 1. Performance analysis for the estimation of faults for Software project: 1.

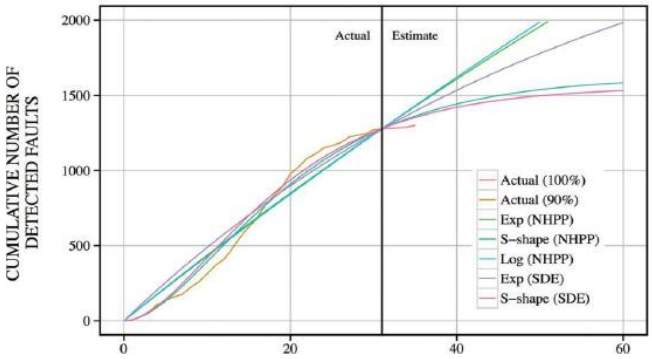

Figure 2. Performance analysis for the estimation of faults forSoftware project: 1.

Table 1. Performance comparison of prediction accuracy for 90% fault data.

Most Suitable	Traditional Network	Convolution Network
NHPP	Exp-SDE	Exp-NHPP
SDE	Exp-SDE	SDE
SDE	Exp-SDE	NHPP

Table 2. Performance comparison of prediction accuracy in percentage for traditional and convolutional networks.

Percentage of Fault Data	Traditional Network RR	Convolutional Network RR
90%	0	68%
80%	67%	83%

RR: Recognition Rate

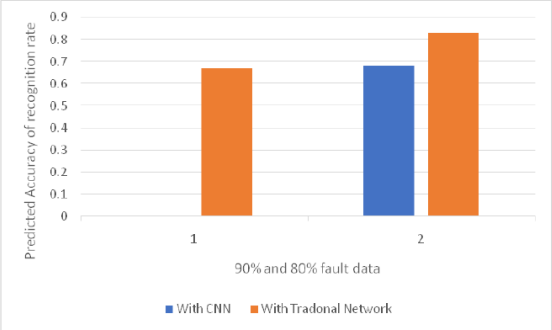

Figure 3. Performance comparison of network for different size of fault data.

VI. Conclusions

This work focuses on proposing a learning model for the optimal selection of SRGM for software development projects. This work aims to design a customized convolutional network for decision making of selecting an appropriate model for the given dataset of faults. This is mainly due to the difficulty in selecting a model based on the past data for the current working projects. So in this paper the convolutional network performance is compared against the traditional neural network and found that the network could able to attain higher accuracy of detecting the faults than earlier network.

References

[1] A. L. Goel and K. Okumoto, Time-dependent error-detection rate model for software reliability and other performance measures, IEEE Transactions on Reliability 28 3 (1979), 3206-211.

[2] P. Roy, G. S. Mahapatra, P. Rani, S. K. Pandey, and K. N. Dey, Robust feed forward and recurrent neural network based dynamic weighted combination models for software reliability prediction, Applied Soft Computing 22 (2014), 629-637.

[3] S. Ramasamy and I. Lakshmanan, Application of artificial neural network for software reliability growth modeling with testing effort, Indian Journal of Science and Technology, Article ID 90093, (9)29 (2016).

[4] P. K. Kapur and R. B. Garg, A software reliability growth model for an error removal phenomenon, Software Engineering Journal 7(4) (1992), 291-294.

[5] S. Yamada and S. Osaki, Software Reliability Growth Modeling: Models and Applications, IEEE Transactions on Software Engineering SE-11, 12 (1985), 1431-1437.

[6] A. L. Goel, Software reliability models: assumptions, limitations, and applicability, IEEE Transactions on Software Engineering 11(12) (1985), 1411-1423.

[7] R. Subburaj, G. Gopal, and P. K. Kapur, A software reliability growth model for estimating debugging and the learning indices, International Journal of Perform ability Engineering 8(5) (2012), 539-549.

[8] R. S. Wahono and N. S. Herman, Genetic feature selection for software defect prediction, Adv. Sci. Lett. 20(1) (2014), 239-244.

[9] H. Wang, T. M. Khoshgoftaar, J. Van Hulse, and K. Gao, Metric selection for software defect prediction, Int. J. Softw. Eng. Knowl. Eng. 21(02) (2011), 237-257.

[10] J. Li, P. He, J. Zhu, and M. R. Lyu, Software defect prediction via convolutional neural network, Proc. IEEE Int. Conf. Softw. Qual., Rel. Secur. (QRS), Jul. 217 318-328.

[11] H. Khanh Dam, T. Pham, S. Wee Ng, T. Tran, J. Grundy, A. Ghose, T. Kim, and C.-J. Kim, A deep tree-based model for software defect prediction, arXiv:1802.00921. 2018.

[12] Karnin, E. D. A simple procedure for pruning back-propagation trained neural networks. IEEE Trans. Neural Netw. 1 (1990), 239-242.

[13] Hutchinson, B., Deng, L., and Yu, D. Tensor deep stacking networks. IEEE Trans Pattern Anal. Mach Intell. 35(2013), 1944-1957.