International Journal on Advances in Engineering Technology and Science
Volume: 5, Issue: 1, 2024, e-ISSN: 2455-3131
DOI: 10.5281/zenodo.10666637

Analysis of Forward Pass RNN with Hyperbolic Tangent Function for Software Defect Prediction

Swati Rai[1], Kirti Jain[2]

[1]Research Scholar, [2]Associate Professor
[1,2]School of Advanced Computing, Sanjeev Agrawal Global Educational University, Bhopal

Abstract– **Software failure prediction and proneness have long been considered critical challenges for the IT industry and software professionals. Conventional approaches may detect software defects inside an application, but they need previous knowledge of problems or faulty components. Automated software fault recovery models enable the programme to significantly predict and recover from software issues via the use of machine learning techniques. This feature reduces mistakes, time, and money while also making the programme run more smoothly. A software defect prediction development model was given using machine learning techniques, which could enable the programme to carry out its intended purpose. A range of optimisation evaluation benchmarks, including as accuracy, f1-measure, precision, recall, and specificity, were also used to evaluate the model's performance. The FPRNN-HTF (Forward Pass RNN with Hyperbolic Tangent Function) deep learning prediction model is based on convolutional neural networks and its hyperbolic tangent functions. The evaluation process showed how well CNN algorithms were used and how accurate they were. Additionally, a comparative metric is used to assess the proposed prediction model in comparison to other approaches. The collected data showed how well the FPRNN-HTF approach performed.**

Keywords– **FPRNN-HTF (Forward Pass RNN with Hyperbolic Tangent Function), precision, recall, specificity, F1-measure, and accuracy.**

I. INTRODUCTION

Software defects have a substantial effect on the reliability, quality, and maintenance costs of the programme. Since most software flaws are hidden, it might be challenging to get bug-free software even with rigorous application. Developing a software bug prediction model that might detect problematic modules early on is also a major difficulty in software engineering. One of the most important tasks in software development is software bug prediction. This is so that by foreseeing the issue modules before software is implemented, user satisfaction and overall programme performance may be raised. Furthermore, early software issue prediction maximises resource efficiency and improves software flexibility to a variety of scenarios.

A lot of research has been done on using machine learning techniques to forecast software defects. Consider the linear Auto-Regression (AR) method for predicting the faulty modules. The study predicts future software problems based on historical data on cumulative vulnerabilities in software. The study also evaluated and contrasted the AR model with the Known Power Model (POWM) using the Root Mean Square Error (RMSE) approach. The study also included three datasets for evaluation, and the results were promising. The study looked at how well a number of machine learning methods worked for predicting defects. The most current advancements in machine learning-based software bug prediction as well as the important prior research on each machine learning technique.

II. BACKGROUND

According to Görkem Giray et al. [1], robotic programming deformity expectation (SDP) strategies are used progressively, sometimes in conjunction with artificial intelligence (AI) methods. However, current machine learning techniques need highlights to be physically removed, which is laborious, time-consuming, and only captures a portion of the semantic information shown in defect describing equipment. Thanks to deep learning (DL) methodologies, professionals have the priceless possibility to extract and profit from more complicated and multi-layered information.

Iqra Batool et al. [2] state that programming engineers may leverage programming issue/deformity expectation to find problematic builds—like modules or classes—early in the product advancement life cycle. Techniques like deep learning, artificial intelligence, and information mining are used to unfulfilled programme expectations.

Haowen Chen et al. [3] established the term "heterogeneous deformity expectation" (HDP), which describes the imperfection prediction amongst projects with varying data. Most existing HDP techniques put source and target data into a traditional measurement space where each piece has no true meaning, significantly limiting their interpretability. In addition, class inequality is a difficulty that HDP often tackles.

Phishing assaults, according to Cagatay Catal et al. [4], try to obtain personal information by using advanced methods, instruments, and strategies. Mobile applications, social engineering, internet forums, and joyful infusion are a few instances of these. Deep learning computations has out to be one of the most successful phishing location strategies, which were created to stop and reduce the dangers of these assaults.

Xieling in addition to others [5], Recent years have seen the emergence of many open-source and endeavor-supported information diagrams, signalling a notable progress in the integration of information depiction and thinking across several fields, such as computer vision and natural language processing. With an emphasis on the topical examination structure, this study aims to comprehensively analyse the condition and trends of information diagrams today.

In Cagatay et al. (2006), novel methods are proposed for recognizing and removing the different kinds of malware, where deep learning computations are essential. Although a lot of effort has been paid to the development of DL-based portable malware detection approaches, it hasn't been completely investigated yet. Finding, gathering, and analyzing published works on the use of deep learning methods to portable malware detection is the aim of this endeavour.

Reality expectation is one of the main obstacles to programming language development and progress in order to further improve programming quality and dependability (Akimova et al., Deform et al., 2007). Finding the contaminated source code exactly and properly is the issue in this field. Creating a prediction model with flaws is a difficult task for which several solutions have been proposed throughout time.

III. PROBLEM IDENTIFICATION

Analytical approaches are often used in development processes for source code verification and examination. This process may be carried either manually or automatically using a variety of tools, including those for static and dynamic code analysis. Recent years have witnessed a boom in the creation of tools for static code analysis, which provides very practical, high-value answers to a wide range of problems faced by software development organizations. However, since there are a lot of false positive and false negative results, these strategies are challenging to use in practical situations. Therefore, a different method or strategy for static code analysis has to be found, such as Machine Learning (ML) methods.

The following is a list of the issues that prior research has revealed:

• Relevant software defects are not always easy to find.

• The recovery of a software problem is not fully understood.

• It may discover the unidentified software issue due to its low accuracy.

IV. RESEARCH OBJECTIVES

The planned work's goals are as follows:

• To improve the precision of faultless software bug recovery.

• To improve recollection for software bugs that affect the retrieval process in every way.

• To improve software bug detection accuracy.

V. METHODOLOGY

The Algorithm of proposed methodology FPRNN-HTF (Forward Pass RNN with Hyperbolic Tangent Function) is as follows

I = Number of input layers

H = Number of hidden layers

O = Number of output layers

S = Number of data set instances

Step 1: for i = 1 to H

Step 2: for j = 1 to S

calculating the forward for the forward hidden layers with activation function

$$h_t^f = \tanh\left(W_h^f h_{t-1}^f + W_x^f x_t + b_h^f\right)$$

end for

Step 3: for j=S to 1

calculating the backward pass for the backward hidden layer's activation function

$$h_t^b = \tanh\left(W_h^b h_{t-1}^b + W_x^b x_t + b_h^b\right)$$

end for

end for

Step 4: for i =1 to O

calculating the forward pass for the output layer using the previous stored activation function

$$P\left(y_t \left| \{x_i\}_{i\neq t}\right.\right) = \sigma\left(W_y^f h_t^f + W_y^b h_t^b + b_y\right)$$

W_y is the weight matrix connecting the hidden layer to output layer,

W_h is the weight matrix that connects hidden to hidden layer,

and W_x is the weight matrix that connects input layer to hidden layer.

b_y is the output layer bias vectors, and b_h is the hidden layer bias vectors.

For the final nonlinearity r, and use tanh as an activation function for classification. According to this form, the RNN will evaluate the output y_t according to the information propagated through the hidden layer regardless of whether it depends directly or indirectly on the values $\{x_i\}_{i=1}^{t} = \{x_1, x_2, ..., x_t\}$.

end for

end for

The Architecture of proposed methodology FPRNN-HTF (Forward Pass RNN with Hyperbolic Tangent Function) is as follows

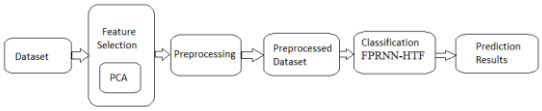

Fig.1- Process of proposed work

VI. RESULTS AND ANALYSIS

The following metrics are collected using a Jupyter notebook with Python 3.11.1 running on Anaconda Navigator. CS1.csv data from the PROMISE dataset is used to calculate precision, recall, F1-Score, and accuracy using the suggested FPRNN-HTF algorithm.

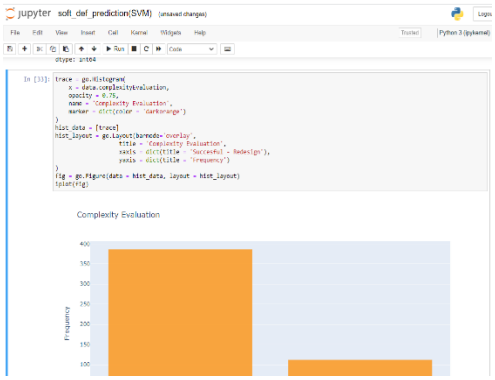

Fig.2- Complexity Evaluation of Bug Frequency for FPRNN-HTF (Proposed Prediction Model)

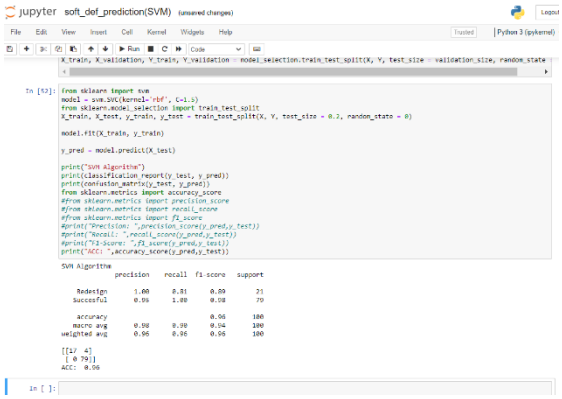

Fig.3- Calculation of confusion matrix, precision, recall, F1-Score and accuracy among different models and FPRNN-HTF (Proposed Prediction Model)

Table 1: Estimation of Precision, Recall, F1-Score and Accuracy among different models and FPRNN-HTF (Proposed Prediction Model)

Models	Precision	Recall	F1-Score	Accuracy
Random Forest	0.9	0.83	0.9	80.65 %
Naïve Bayes	0.94	0.82	0.88	80.10 %
Logistic Regression	0.93	0.8	0.88	79.5 %
Decision Tree	0.83	0.84	0.83	74.86 %
ANN	0.92	0.84	0.9	82.77 %
FPRNN-HTF (Proposed)	**0.95**	**0.96**	**0.98**	**96 %**

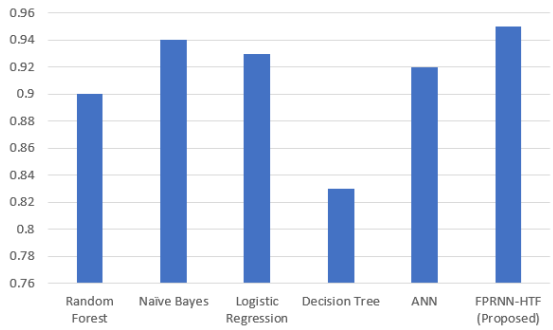

Fig.4- Graphical Analysis of Precision among different models and FPRNN-HTF (Proposed Prediction Model)

In the context of bug prediction, the accompanying image illustrates that the suggested model delivers more accuracy when compared to other models. FPRNN-HTF outperforms Naive Bayes by a margin of 0.01 in terms of accuracy.

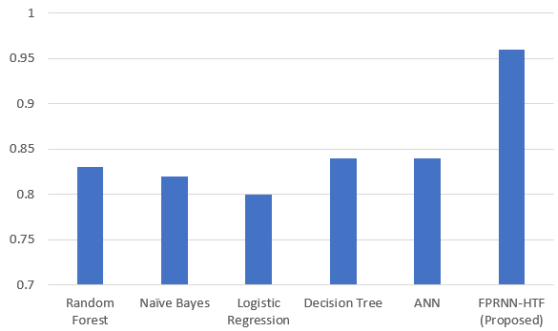

Fig.5- Graphical Analysis of Recall among different models and FPRNN-HTF (Proposed Prediction Model)

The suggested model performs better than previous models in terms of recall for bug prediction, as shown in the graph above. Recall for FPRNN-HTF is 0.12 times higher than that of Decision Tree and ANN prediction models.

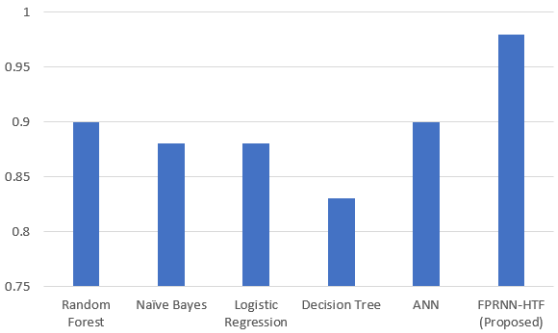

Fig.6- Graphical Analysis of F1-Score among different models and FPRNN-HTF (Proposed Prediction Model)

The suggested model's higher F1-score compared to previous models is seen in the following graph. FPRNN-HTF outperforms Random Forest and ANN by 0.08 points in terms of F1-score.

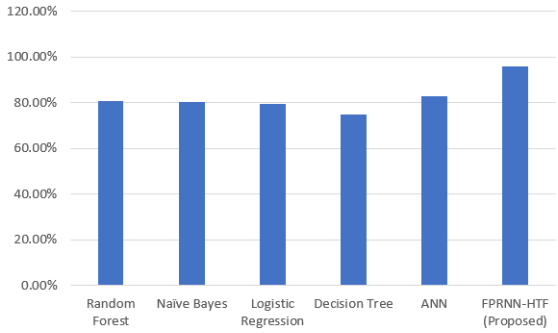

Fig.7- Graphical Analysis of Accuracy among different models and FPRNN-HTF (Proposed Prediction Model)

The following data demonstrates that the suggested model provides a higher level of prediction accuracy for defects when compared to the existing models. Compared to the ANN prediction model, the FPRNN-HTF prediction model has a 13.23% higher accuracy.

VII. CONCLUSION

To get the desired results in this experiment, we used the FPRNN-HTF model. Our investigation shows that feature selection and cross validation were not given enough consideration in previous efforts. On big datasets, the suggested method performs better than others in terms of accuracy (98.16%). Since it is computationally intensive (it prevents overfitting and provides quick prediction speeds) and adaptable in application (it can be used for both regression and classification issues), the combination of the five proposed algorithms produces the best results. It has been an ongoing project to look at this technique further for bug prediction in deep learning models.

These should lead to certain conclusions:

1. The suggested model has a greater accuracy than FPRNN-HTF. If we compare this accuracy to Naïve Bayes, it has risen by 0.01.

2. Recall is greater with the suggested model than with FPRNN-HTF Regression. Recall for FPRNN-HTF is 0.12 times higher than that of Decision Tree and ANN prediction models.

3. The proposed model outperforms the FPRNN-HTF in terms of F1-Score. There is a 0.08 difference between Random Forest and ANN.

4. The accuracy of the suggested model is higher than that of ANN. A 13.23% improvement in accuracy has been seen.

Therefore, FPRNN-HTF (Forward Pass RNN with Hyperbolic Tangent Function) is a more accurate approach for software bug prediction.

We suggest a method that improves diagnosis accuracy, which is essential for a successful course of therapy. Future accuracy assessments should make use of fresh datasets, and further AI techniques have to be used to verify the estimate's accuracy. The recommended model has a processing time constraint since the train data performance estimate requires a huge quantity of data. In the future, the same algorithms and real-time data will be used to measure the system's efficacy.

REFERENCES

[1] Görkem Giray, Kwabena Ebo Bennin, Ömer Köksal, Önder Babur, Bedir Tekinerdogan, "On the use of deep learning in software defect prediction", The Journal of Systems & Software, 2023.

[2] Iqra Batool, Tamim Ahmed Khan, "Software fault prediction using data mining, machine learning and deep learning techniques: A systematic literature review", Computers and Electrical Engineering, May 2022.

[3] Haowen Chen, Xiao-Yuan Jing, Yuming Zhou, Bing Li, Baowen Xu, "Aligned metric representation based balanced multiset ensemble learning for heterogeneous defect prediction", Information and Software Technology, July 2022.

[4] Cagatay Catal, Görkem Giray, Bedir Tekinerdogan, Sandeep Kumar & Suyash Shukla, "Applications of deep learning for phishing

detection: a systematic literature review", Knowledge and Information Systems , 2022.

[5] Xieling Chen, Haoran Xie, Zongxi Li, Gary Cheng, "Topic analysis and development in knowledge graph research: A bibliometric review on three decades", Neurocomputing, October, 2021.

[6] Cagatay Catal, Görkem Giray, Bedir Tekinerdogan, "Applications of deep learning for mobile malware detection: A systematic literature review", 2021.

[7] Konstantin S. Kobylkin, Anton V. Konygin Ilya P. Mezentsev and Vladimir E. Misilov, "A Survey on Software Defect Prediction Using Deep Learning", IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), 2021.

[8] Farah Atif, Manuel Rodriguez, Luiz J. P. Araújo, Utih Amartiwi, Barakat J. Akinsanya & Manuel Mazzara "A Survey on Data Science Techniques for Predicting Software Defects", IEEE Conf. of Software Engineering, 2021.

[9] Saleema Amershi; Andrew Begel; Christian Bird; Robert DeLine; Harald Gall; Ece Kamar; Nachiappan Nagappan, "Software Engineering for Machine Learning: A Case Study", IEEE Conf. on Machine Learning, 2019.

[10] George G. Cabral; Leandro L. Minku; Emad Shihab; Suhaib Mujahid, "Class Imbalance Evolution and Verification Latency in Just-in-Time Software Defect Prediction", IEEE/ACM 41st International Conference on Software Engineering (ICSE), 2019.

[11] Kwabena Ebo Bennin; Jacky Keung; Passakorn Phannachitta; Akito Monden; Solomon Mensah, "MAHAKIL: Diversity Based Oversampling Approach to Alleviate the Class Imbalance Issue in Software Defect Prediction", IEEE Transactions on Software Engineering, 2018.

[12] Yuxiang Gao, Yi Zhu, Yu Zhao, "Dealing with imbalanced data for interpretable defect prediction", IEEE Conf on Data Analysis, 2017.

[13] Kwabena Ebo Bennin; Jacky Keung; Akito Monden; Yasutaka Kamei; Naoyasu Ubayashi, "Investigating the Effects of Balanced Training and Testing Datasets on Effort-Aware Fault Prediction Models", IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), 2016.

[14] Faruk Arar, Kürşat Ayan, "Software defect prediction using cost-sensitive neural network", Applied Soft Computing, Volume 33, August 2015.

[15] Deepika Badampudi, Claes Wohlin, Kai Petersen, "Experiences from using snowballing and database searches in systematic literature studies", 19th International Conference on Evaluation and Assessment in Software Engineering, 2015.

[16] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation", Computation and Language, 2014.

[17] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique", Artificial Intelligence, 2011

[18] Gul Calikli; Ayse Tosun; Ayse Bener; Melih Celik, "The effect of granularity level on software defect prediction", 24th International Symposium on Computer and Information Sciences, 2009.

[19] Cagatay Catal, Banu Diri, "A systematic review of software fault prediction studies", Expert Systems with Applications, Volume 36, Issue 4, May 2009.

[20] Alessandro Birolini, "Reliability and Availability of Repairable Systems", Reliability Engineering, 2004.