Available online at: http://www.advancedscientificjournal.com
http://www.krishmapublication.com
IJMASRI, Vol. 1, issue 1, pp. 58 - 61, Apr. -2025
https://doi.org/10.53633/ijmasri

INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY ADVANCED SCIENTIFIC RESEARCH AND INNOVATION (IJMASRI)

ISSN: 2582-9130 **IBI IMPACTFACTOR 1.5** **DOI: 10.53633/IJMASRI**

RESEARCH ARTICLE

A HYBRID DEEP LEARNING APPROACH WITH GENETIC AND CORAL REEFS METHEURISTICS FOR ENHANCED DEFECT DETECTION IN SOFTWARE

Mrs. P.Kalaiselvi

Head and Assistant Professor , DepartmentofinformationTechnology, Achariya Arts science college, Puducherry
Email: pkalaiselvimphil@gmail.com

Abstract

In the software development process, it is crucial to identify and fix software flaws early on. Defective software has a detrimental effect on operating expenses throughout the production phase, which in turn affects customer satisfaction. While there are various methods for forecasting software flaws, prompt and precise detection are two crucial components. A hybrid Deep Neural Network model for improved software bug prediction is presented in this paper. In order to optimize the Deep Neural Network design, many nature-inspired algorithms have been used to enhance the exploration of the hyper parameter solution space. Software faults have been experimentally investigated using NASA datasets, and performance comparisons have been made using evaluation metrics like as accuracy, computing time, and F1 score. With an average F1-score of 0.92 and the greatest accuracy of about 96%, the method based on the combination of the Genetic and Coral Reef metaheuristics performed better than any other model.

Keywords: Predicting software bugs Deep learning Networks of neurons An algorithm that uses genetics The methodfor optimizing coral reefs Optimization methods inspired by nature Metaheuristic algorithms

Introduction

The software market has grown steadily as a result of the widespread use of cutting-edge modern technologies across many industries. Even though these inventions have greatly increased the productivity of our daily tasks, they also carry the risk of seriously up setting our contemporary economies, particularly if they fail. Even if there are a lot of effective software systems and solutions on the market today, a general lack of effort and quality control will lead to longer production times, overspending, missed deadlines, and eventually a decline in customer loyalty.

Literature Review

Review of Literature Predicting software bugs has gained importance since 1990, and as software engineering has advanced, so too has the number of research articles published in this area. The vast majority of applications have flaws. These fall into one of four categories: Critical, High, Medium, or Low. Theymaybe caused by mild or major problems (Dhavakumar & Gopalan, 2021).

Supervised Learning

Learning Under Supervision According to Perreault et al. (2017), a variety of supervised learning algorithms, including Naïve Bayes, NN, Linear Regression, Support Vector Machines (SVM), and K-nearest neighbor, have been utilized for software bug prediction and detection. An efficient prediction model has been constructed using Naïve Bayes with feature selection (Shivaji et al., 2013). However, because of the training network's undiscovered nodes and arcs, Naïve Bayes was discovered to be the primary source of performance degradation (Dejaeger et al., 2013). Numerous tests have also been carried out to evaluate the feasibility of Support Vector Machines (SVM) for bug prediction because of its capacity to handle non-linear data utilizing kernel functions (Kavzoglu and Colkesen, 2009).

Un supervised Learning

Learning Without Supervision Semi-supervised classification methods have been proposed by many studies (Bennett and Demiriz, 1999; Jaochims, 1999; Belkin et al., 2006). However, the majority of researchers have addressed class imbalance difficulties (Lessmann et al., 2008), and only a small number have concentratedonboth labeled and unlabeled datasets at the same time (Chawla and Karakoulas, 2005; Arshad et al., 2018).Lu et al.

DNN and Hybrid Models

Hybrid and DNN Models In order to define the relationship between software metrics and software bug proneness, Jin and Jin (2015) suggested a hybridized model for bug prediction that uses NN and a variation of Particle Swarm Optimization (PSO). ANN was used to classify modules as either bug-proneor non-bug-prone, and QPSO (Quantum Particle Swarm Optimization) was utilized to pick features. Similarly, Rathore and Kumar (2015) used genetic programming to create a model that uses the frequency of flaws in software to address quality issues. Recall, error, and completeness were used to assess the prediction model's performance. This suggested genetic programming model demonstrated a high degree of accuracy in anticipating a variety of problems. Using Deep Learning to extract metrics to train the classifier to identify defects,Yangetal. (2015) created a technique for just-in-time defect detection.

Methodology

Machine learning Approach

Decision-making in policy analysis has benefited from the application of machine learning approaches to big data sets, particularly when predicting results based on historical data (Chand and Zhang, 2022).

Based on Deep Neural Networks (DNN), our suggested method allows us to:

I. use the Ant Colony Optimization (ACO), Bat (BA), Particle Swarm Optimization(PSO), Cuckoo Seach (CS), Firefly (FA), and Coral Reefs Optimization (CRO) algorithms, among other Nature-Inspired Algorithms (NIAs), to optimize the hidden layer configurations.

II. To get even more optimization, combine the NIAs listed above with Genetic Algorithms (GA).

Deep Learning

The Artificial Intelligence family includes the Deep Learning structure, commonly referred to as representation learning. A number of neurons joined in layers make up the DNN structure. Weights ranging from 0 to 1 are used to transport all relevant data through the levels. Together with additional input known as bias, which has a value of 1, the neurons in

the layers carry out an addition. The smallest level necessary for a neuron to activate is represented by the bias value, which is a threshold. After evaluating the summation result, the information is transmitted to further neurons that are connected to it until the final layer is reached. Next, the DNN's output is acquired. In order to guarantee optimal training and avoid over fitting of the model, early stopping is employed. Additionally, dropout is implemented at the first layer to allow for equal neuron training by randomly removing neurons at each epoch, preventing over fitting. Only the highest accuracy has been reported for this investigation, despite the consideration of several criteria.

Deep Neural Network (DNN) with Nature-Inspired Algorithms (NIAs) Initializing a Deep Neural Network (DNN) with random weights and biases is a common step in training the DNN with any metaheuristic optimization strategy. The intended network receives the datasets. The root mean square error (RMSE) error function isused to calculate the error value and provide the fitness value. The next set of weights and biases for the iteration is generated by applying the metaheuristic approach with random values. Until the termination requirement is met, this procedure keeps going. The solution with the fewest errors at the end of all iterations is considered the best one.

GeneticAlgorithms

One popular metaheuristic algorithm that draws inspiration from biological evolution is the genetic algorithm (GA). GA is based on biologically inspired operators, fitness selection, and chromosome representation (Michalewicz and Schoenauer, 1996). Recently, GA-based research has been applied to NN models to address computational cost, ease of hardware implementation, and better accuracy. Numerous scholars from other fields, including as manufacturing processes, financial markets, energy consumption, medical diagnosis, and cloud resource utilization, have been captivated by this approach (Whitely et al. 1990). Nonetheless, the study identified a few of Gas performance limitations, such as significant performance degradation and early convergence that impairs both the system's functionality and search capabilities (Ramesh and Krishnan, 2009). The network parameters used in this GA analysis were taken from the Arroyo, February 2022 study.

Nature-Inspired Algorithms (NIAs) - Genetic Algorithms (GA) based Deep Neural Network (DNN)

To enhance the functionality of conventional neural networks, researchers have resorted to GA(Asadniaetal.,2014). To create GA, three evolutionary processes are needed: selection, crossover, and mutation. According to experiments, the GA's convergence time is significantly slowed down when large training samples areused (Soleymani Yazdietal., 2010). Additionally, applying GA's crossover function to neural networks may result in the "permutation problem." Because of this, applying GA is frequently seen as a challenging procedure. Another significant problem with conventional DNNs is over fitting. If there are some what more weights than data sets for the DNN's training, "over fitting" could happen.

Proposed Optimization Approach

The suggested model uses k - folds cross- validation and arrange of NIAs to solve the issue of parameter setting in DNN for software bug identification. Creating about equal-sized sets from all of the data items is the aim of k-fold cross-validation. To calculate the test set's parameters, a 10-fold cross-validation method is used. Ten subsets, each of the same size, are randomly selected from each data set; nine of these subsets are used as training data, while one is always utilized as test data. We evaluated the model with k = 3, k = 5, and k = 5. With k=10, the highest accuracy was achieved. Because stratified 10-fold cross validation has become the industry standard in terms of practicality, it has been adopted. According to certain research, stratification can also marginally improve performance (Written et al., 2011).

Discussions

In the software development and maintenance lifecycles, defect prediction is a crucial subject that influences the entire performance of software products. The goal of this study's hybrid models was

to identify the method that best optimizes the categorization of software problems when paired with a deep neural network. The studies' findings show that the Coral Reefs Optimization (CRO) algorithm and the Cuckoo Search (CS) algorithm in conjunction with the Genetic algorithm are the most accurate models. With the best accuracy of 96.67% and average F1-score of 0.92, the DNN-GA-CRO was the most successful. DNN-GA-CS came in second with the highest accuracy of 93.78% and average F1- score of 0.90. The DNN performs significantly worse when the Nature-Inspired meta-heuristic techniques are used separately. This is primarily due to their in ability to come up with a suitable stopping criterion on their own. Therefore, either the maximum number of it erations or the achievement of a specific error value determine this. However, a more robust strategy for convergence to the global optimum and avoiding local-minima traps is brought about by combining the evolutionary method of the Genetic Algorithm with the characteristics of each meta-heuristic. The Genetic Algorithm will repeatedly search for the ideal parameters until the stop criterion is satisfied after the Nature Inspired Algorithm selects the best population (including weights and biases). The Coral Reefs Optimization (CRO) algorithm, a cellular evolutionary method, has shown remarkable convergence properties in achieving a global optimal value.

Conclusion and Future Work

To improve the quality of soft ware products, it is necessary to detect software flaws early and accurately, not withstanding their complexity. According to this study, hybrid prediction models that include DNN and NIAs have proven to be efficient and capable of producing the best neural network parameters, which improves classification accuracy and prevents local minima. The PROMISE repository's datasets were used to assess the performance of the various suggested models. To prevent erroneous interpretations, the dataset required a great deal of pre-processing. Our experimental findings demonstrate that the hybridization of DNN with the Genetic and Coral Reefs algorithms worked better than the alternative technique, resulting in a notable improvement in prediction accuracy.

References

1. Demiriz, A., and Bennett, K. (1999). support vector machines that are semi-supervised. Neural information processing system advancements, 11,368–374.
2. Diri, B., and C. Catal (2009). Acomprehensive analysis of software failure prediction. 7346–7354 in J. Expert Syst.Appl..36.
3. Jin, Shu-Wei, and Jin, Cong (July 2015). A method for predicting software fault- proneness that combines quantum particle swarm optimization with hybrid artificial neural networks. 35, 717–725; Applied Soft Computing 10.1016/j.asoc.2015.07.006. A DOI.
4. DiWu, Guoyou Zhang, XingjuanCai, Shaojin Geng, Jiangjiang Zhang, A Multi-objectiveBat Algorithm for Predicting SoftwareDefects,Springer2020inSingapore Google (2015) Terms of service for Google. https://www.tensorflow.org is where it is accessible.(Received:May11,2020)Saha, A., and Shani Arora (2018). A comparison of support vector machines and artificial neural networks for software defect prediction.DOI:10.1007/978-981-10-4603.

******