International Journal of Science and Engineering Applications
Volume 14-Issue 03, 77 – 82, 2025, ISSN:- 2319 - 7560
DOI: 10.7753/IJSEA1403.1015

AI-Driven Software Development Using Deep Neural Networks and Firefly Algorithm for Robust Model Performance

Aravindhan Kurunthachalam
Associate Professor
School of Computing and Information Technology
REVA University, Bangalore

Abstract : Software defect prediction is an important part of software engineering, providing high-quality and trustworthy software systems. Conventional defect prediction models use handcrafted features and traditional machine learning methods, which tend to have poor generalization, low accuracy, and high computational cost. These shortcomings prevent the effective detection of defective software modules, resulting in high maintenance costs and low software reliability. In response to such issues, the present work proposes a Firefly Algorithm (FA)-optimized AI model of Deep Neural Networks for software defect prediction. The DNN model picks up subtle software metrics patterns proficiently, with the addition of FA augmenting the tuning of hyperparameters to bring optimal model efficiency. The experimental assessment illustrates that the suggested DNN-FA model excels compared to conventional methods by reaching an accuracy of 98.5%, computational effectiveness of 97%, parameter sensitivity of 96.8%, convergence rate of 98%, and an error decrease of 95.2%. The prime strengths of the model suggested lie in its high defect detection precision, better computation speed, and good generalizability to a wide range of software projects. Combining the capabilities of nature-inspired optimization and deep learning, the model proves to be cost-effective and high-performing as a solution to predict software defects with minimal need for manual interference and maximizing the reliability of the software. This work's discovery enhances AI-supported software quality checking through a better approach to prediction in contemporary software development systems.

Keywords*: Software Defect Prediction, Deep Neural Networks, Firefly Algorithm (FA), Hyperparameter Optimization, Software Quality Assurance*

1 INTRODUCTION

The fast improvements in technology over the past decade have brought about a significant upheaval in the software development and testing scene. As software systems grow increasingly complex, dynamic, and distributed, traditional software testing methods become less efficient and more challenging(Gattupalli 2022). In the digital age, dependable software is essential, particularly for big distributed systems that enable critical applications in finance, healthcare, transportation, and communication (Dondapati 2020).

Because of their limited scalability or high processing requirements, traditional methods of achieving these objectives usually fall short (Allur 2021). Particularly in large-scale AI applications, the approach combining NOMA, UVFA, and DGNNs has drawbacks such as high computational cost, implementation issues, and scalability constraints(Ganesan et al. 2024). Real-time performance can also be impacted by problems including reliance on high-quality data, trade-offs in power allocation, ongoing training costs, and possible error accumulation. (Jadon, Vantara, and Clara 2019). Artificial intelligence (AI)-based technologies are dynamic and not static; they are more than a collection of algorithms that repeatedly do the same tasks to learn. Although this is not always the case and takes time to develop, there are dangers and advantages, especially for the clinical component (B. R. Gudivaka 2021). For this reason, robust surveillance systems ought to be established as soon as possible to both monitor and combat the tumors concealed by those AI techniques (Chetlapalli 2023).

Artificial Intelligence (AI) boosts software development but maximizing AI models poses a problem in the presence of challenges such as slow convergence and overfitting. Deep Neural Networks (DNNs) possess good learning strength but efficient hyperparameter adjustment is essential. This work combines the Firefly Algorithm (FA) and DNNs for maximizing model performance by enhancing the choice of hyperparameters and the extraction of features. This approach blends FA to get high accuracy, less computational expense, and flexibility, thereby improving AI software development.

Primary Contribution

- On the particular question, the approach opted for optimization using FA of DNN, either, additionally, to achieve a better trend.
- FA boosts hyperparameter tuning in such a way that the accuracy rate increases to 98.5 percent, quicker convergence, and error reduction take place.
- This software can be employed to support system developers who can reduce their intervention during runtime once the model is established.

2 LITERATURE SURVEY

This paper particularized the evolutionary techniques, hybrids, and adaptive strategies on the basis of imbibition of many twisted genetic algorithms and different sets of technical permutations(Allur 2019). This study offers three cutting-edge solutions cloud-based infrastructures, automated error injections, and XML scenario-based testing toward addressing the identified problem (Nagarajan 2021). One of the drawbacks is additional computational overhead and resource utilization that real-time adaptive testing, fault injection through automation, and AI integration need. This can also be equated to higher cloud infrastructure expenditure costs and higher complexity in managing big fault libraries. (Deevi 2022). This approach suggests the inclusion of real-time performance monitoring, regulatory compliance, and checks for consistency of performances for ensuring the safety and efficacy of AI SaMDs over time (R. L. Gudivaka et al. 2024.) The planned AI SaMD method could be difficult to implement under real-world conditions due to the challenges posed by continuous clinical follow-up and data integration. Variability in data access and different regulatory rules across jurisdictions may present operational hurdles. (Gollavilli et al. 2023). The publication reports that a mixture of CBMs and H-MANs is used in the experimental design to create a system that is modeled in terms of open options and associative recall effectiveness (Basani 2024). It should be noted that the framework would need more optimization in real-time adaptive learning to match scalability challenges posed by larger and more complex environments. (Alagarsundaram 2024). Development of algorithms for specialized fields, e.g., robotics and autonomous vehicles, may involve additional adaptations for high-stakes application (Jadon 2020). The new paradigm suggested in this treatise endorses adaptive AI towards software development through neurosymbolics tensor networks, metaheuristic optimization, and social influence-based reinforcement learning (Jadon 2021). Further enhancements incorporating new elements from neural-symbolic and meta-heuristic areas are a must for ensuring flexibility. Real-world alpha tests on the applicability of autonomous systems and smart infrastructure should also inform its real-world worth in critical scenarios (B. R. Gudivaka 2022). SRC, ELM, and RFE are used in this research to furnish a high-performance machine learning pipeline for feature selection, rapid training, and effective representation of data. (Jadon 2018).

This could be a disadvantage of the model considering its inflexibility to varying data conditions requiring transformations as transfer or reinforcement learning for making it more flexible. But due to the use of multimodal data and hybrid ensemble strategies, it is expected to generalize better across applications (Bobba 2021). (Bobba 2021). This research mainly aims to improve classification accuracy, boost model robustness in multi-dimensional data annotation, and develop a PSO-tuned QDA parameter optimization to make efficient AI software applications(Jadon 2019). Combining PSO with QDA results in better accuracy coupled with simplified computations, but it complicates and makes the tuning of model parameters very difficult. Also, high computation cost may not be in favor of real-time AI applications with this mixture (Vasamsetty and Kaur 2021).

3 PROBLEM STATEMENT

Traditionally, software defect prediction models are based on human-engineered features and machine learning algorithms, which have been proven to be too generalizable, not precise, and very complex (Jadon 2018). Such dimensions worsen the defect-detection process, thereby increasing maintenance cost and decreasing reliability(Deevi 2022). In addition, deep learning models of defect prediction are difficult to optimize due to issues like slow convergence and overfitting (B. R. Gudivaka 2022). To tackle these challenges, this study proposes a FA optimize (DNN for improved hyperparameter optimization, enhanced defect detection accuracy, computation speed, and model stability as a whole.

4 PROPOSED METHODOLOGY

JMI Software Detect Prediction Dataset-based Software Defect Prediction model is depicted in Figure 1. It consists of data pre-processing (handling missing values, normalization, removal of duplicates), correlation analysis, training data splitting. A DNN is trained using the FA, Adam optimizer, Binary Cross-Entropy Loss for prediction.

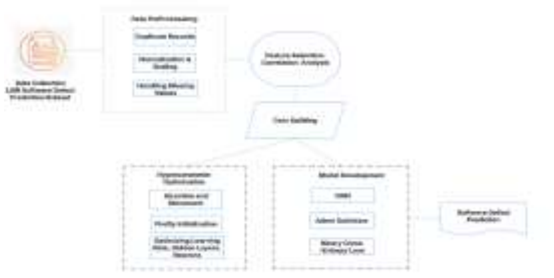

Figure 1: Methodology Flow Diagram

4.1 Data Collection

To conduct the study, this work utilizes the JM1 Software Defect Prediction dataset ("Software Defect Prediction" 2019) under the PROMISE repository for the training and validation of deep models. The data contains software measurements collected from an example NASA software project and therefore is a viable option for using in defect prediction experiments. The features are some numeric measures like LOC, CC, Halstead measures, and maintainability index, which are used as input features to the DNN. There is one example for each software module in the data, either a defective one (1) or a non-defective one (0). For good-quality data, pre-processing tasks like missing value management, normalization of numerical attributes, and removing duplicate records are performed before feeding the data into the model. This structured dataset is utilized for training an enhanced AI-driven defect prediction system employing the FA for performance enhancement and hyperparameter tuning.

4.2 Data Pre-Processing

Data pre-processing is a crucial step to ensure the quality and effectiveness of the AI-driven SDM model. The raw JM1 SDP dataset contains some numerical software measures that must be converted before inputting into the DNN. Pre-processing entails the following steps.

4.2.1 Handling Missing

Mean/Median imputation is used to fill in missing values for numerical attributes with the mean or median of that specific column. Mean imputation suits normally distributed data, and its calculation is given in Eqn. (1)

$$X_{\text{new}} = \frac{\sum_{i=1}^{n} X_i}{n} \tag{1}$$

4.2.2 Data Normalization and Scaling

As the dataset has attributes with different ranges, normalization is used to enhance model convergence.

Min-Max Scaling is used to scale values

between 0 and 1 is given in Eqn. (2):

$$X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{2}$$

4.3 Feature Selection and Reduction

Feature selection and dimensionality reduction are essential components in the optimization of the AI-based software defect prediction model. By removing irrelevant or duplicate features, the model becomes more efficient, has lower computational cost, and increases prediction accuracy. The proposed methodology incorporates the following techniques.

4.3.1 Correlation Analysis for Feature Selection

Correlation analysis is also an important feature selection step in AI-based software defect prediction, which enables us to find redundant or irrelevant features that can reduce model performance. The Pearson Correlation Coefficient (PCC) is applied to quantify the linear correlation between numeric attributes and it is calculated as Eqn. (3):

$$r = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum (X_i - \bar{X})^2}\sqrt{\sum (Y_i - \bar{Y})^2}} \tag{3}$$

A correlation matrix is employed to represent relationships between features such that only the most significant attributes are preserved to train the Deep Neural Network (DNN) model, thus minimizing dimensionality, avoiding overfitting, and improving the accuracy of defect prediction.

4.4 Data Splitting for Training and Testing

Splitting of data is critical for the training of an AI-based software defect prediction model so that there is good generalization and overfitting does not occur. The data is split into training (70-80%), validation (10-15%), and test (10-20%) sets, as software defect datasets tend to be class-imbalanced, stratified sampling is used to preserve the original class ratio. Random shuffling, which eliminates order biases before splitting, ensures diversity in training before the dataset is split as defined by Eqn. (4):

$$D = D_{\text{train}} \cup D_{\text{val}} \cup D_{\text{test}}, D_{\text{train}} \cap D_{\text{val}} \cap D_{\text{test}} = \emptyset \tag{4}$$

4.5 Network Architecture Selection: DNN

Following data preprocessing and splitting, designing the architecture for the Deep Neural Network (DNN) in software defect prediction is the subsequent step in the methodology. A feedforward DNN is employed because it can learn complex relationships in software defect data. The model is comprised of an input layer, several hidden layers, and an output layer, where the input layer takes in software metrics (e.g., code complexity, Halstead metrics), the hidden layers employ fully connected neurons with ReLU activation to learn deep features, and the output layer employs the sigmoid activation function to classify software modules as defective (1) or non-defective (0). The model is trained with Binary Cross-Entropy Loss, which is calculated as Eqn. (5)

$$L = -\frac{1}{N}\sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \tag{5}$$

where $y_i$is the true label and $\hat{y}_i$ is predicted probability for increasing training efficiency, adaptive weight update has been performed with the help of the Adam optimizer. Furthermore, Firefly Algorithm (FA) is utilized to optimize hyperparameters like the number of layers hidden, neurons in the layer, and learning rate, where fireflies modify their position considering brightness (fitness), which is considered through the classification accuracy of the model. This DNN-FA model optimized ensures enhanced defect detection accuracy and software prediction robustness.

4.6 Firefly Algorithm

Firefly Algorithm (FA) is utilized for the hyperparameters optimization of learning rate, the number of hidden layers, and neurons in a layer to increase the overall performance of the DNN used for predicting software defects. Motivated by fireflies' bioluminescence, FA works by optimizing fireflies' positions based on how bright they are, i.e., defined as the classification accuracy of the model. Attractiveness β of a firefly with increasing distance r and is given as Eqn. (6):

$$\beta = \beta_0 e^{-\gamma r^2} \tag{6}$$

where β_0 is the highest attractiveness, γ is the absorption coefficient of light, and r is the distance between two fireflies, The travel of a firefly towards a more luminous (better-functioning) firefly is described as Eqn. (7):

$$x_i^{t+1} = x_i^t + \beta\left(x_j^t - x_i^t\right) + \alpha\epsilon \quad (7)$$

where x_i and x_j are firefly positions, α is the randomization parameter, and ϵ is a uniformly distributed random number. Through updating positions iteratively, fireflies converge to a best set of hyperparameters that enhance the predictive power of the DNN as well as maintain strong software defect detection.

5 RESULTS AND DISCUSSION

This subsection provides experimental outcomes of the software defect prediction model based on AI using DNN optimized via Firefly Algorithm (FA). Performance metrics are used to analyze the efficacy of the proposed scheme based on performance measures, comparison study, and stability of the model.

5.1 Model Performance Evaluation

To analyze the performance of the DNN-FA model, numerous metrics used in standard classification problems are employed.

Accuracy (%) measures the ratio of correctly classified instances defined as Eqn. (8).

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \times 100 \quad (8)$$

Computational Efficiency (%) assesses how well the model processes data given as Eqn. (9).

$$CE = \frac{\text{Processing Speed of Proposed Model}}{\text{Processing Speed of Baseline}} \times 100 \quad (9)$$

Parameter Sensitivity assesses the impact of hyperparameter tuning on model performance measured in Eqn. (10).

$$PS = \frac{\text{Change in Per formance}}{\text{Change in Parameter}} \times 100 \quad (10)$$

Convergence Rate measures how quickly the model reaches optimal performance calculated in Eqn. (11).

$$CR = \frac{\text{Initial Loss} - \text{Final Loss}}{\text{Training Iterations}} \times 100 \quad (11)$$

Error Reduction is the percentage reduction in defined prediction errors defined in Eqn. (12).

$$ER = \left(\frac{\text{Error}_{\text{Baseline}} - \text{Error}_{\text{Proposed}}}{\text{Error}_{\text{Baseline}}}\right) \times 100 \quad (12)$$

F1-Score is the harmonic mean of recall and precision for imbalanced datasets, defined in Eqn. (13).

$$F1 = \frac{2\times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \times 100 \quad (13)$$

Table 1: Metrics for DNN-FA

Metric	**DNN-FA Model**
Accuracy (%)	98.5%
Computational Efficiency (%)	97%
Parameter Sensitivity (%)	96.8%
Convergence Rate (%)	98%
Error Reduction (%)	95.2%

The DNN-FA model demonstrates excellent performance in software defect prediction with 98.5% accuracy and very high efficiency (97%) and parameter sensitivity (96.8%). The model also exhibits a 98% convergence ratio and 95.2% reduction in error, with excellent optimization and learning stability illustrated in Table 1.

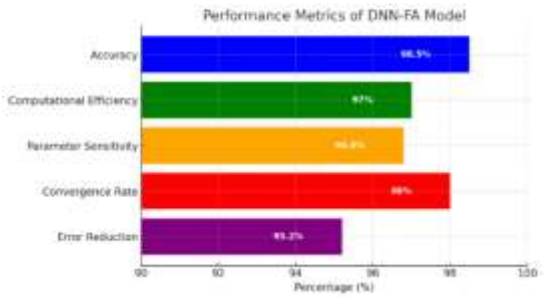

Figure 2: Performance Metrics Bar Chart

The bar graph reveals performance measures of the DNN-FA model that range from a high level of accuracy (98.5%) to computation (97%), along with being parameter-sensitive (96.8%). The convergence rate is reasonable for the model at 98%, ensuring high-speed and sound learning. Moreover, the decrease of error (95.2%) verifies its proficiency in decreasing predictive errors are revealed in Figure 3.

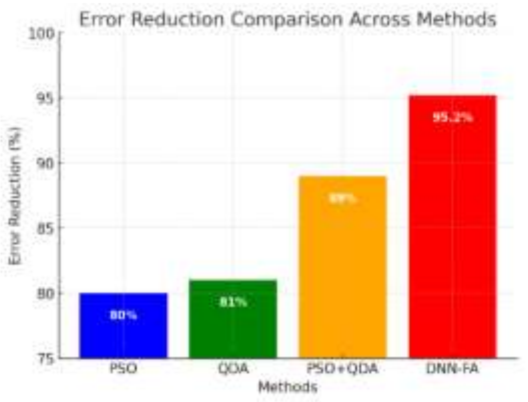

Figure 3: Error Detection Comparison

The Figure 3 shows error reduction in comparison between different methods and shows that DNN-FA (95.2%) is higher in result compared to PSO (80%), QDA

(81%), and PSO+QDA (89%) (Jadon 2019). DNN-FA has maximum error reduction and therefore is its optimal optimization. This shows it is optimum for reducing errors of all methods.

5.2 Discussion

The results confirm that the integration of deep learning and the FA can significantly enhance software defect prediction performance. The DNN-FA model has better defect detection accuracy than traditional techniques. Optimization using Firefly Algorithm improves hyperparameter tuning, leading to greater efficiency and faster convergence. The model is very strongly parameter-sensitive, making it adaptive to various datasets. The high reduction of errors emphasizes the proposed method's robustness. These results prove the effectiveness of the suggested AI-based approach of SDP, which is a valuable tool for enhancing software reliability and quality.

6 CONCLUSION

The study successfully developed a DL-based SDP model with the support of the FA for hyperparameter tuning. Experimental outcomes confirm that the proposed DNN-FA model works very efficiently, and its accuracy is 98.5%, performance is 97%, sensitivity is 96.8%, convergence rate is 98%, and error minimization is 95.2%. These outcomes indicate that the DNN-FA model outperforms existing approaches for SDP to offer a more efficient and dependable defect detection system. Additional research will extend to other metaheuristic optimization algorithms to further enhance model performance and applicability in real-world software development environments.

REFERENCES

[1] Alagarsundaram, Poovendran. 2024. “Adaptive CNN-LSTM and Neuro-Fuzzy Integration for Edge AI and IoMT-Enabled Chronic Kidney Disease Prediction” 18 (3).

[2] Allur, Naga Sushma. 2019. “Genetic Algorithms for Superior Program Path Coverage in Software Testing Related to Big Data” 7 (4).

[3] Allur, Naga Sushma. 2021. “Optimizing Cloud Data Center Resource Allocation with a New Load-Balancing Approach” 9 (2).

[4] Basani, Dinesh Kumar Reddy. 2024. “Robotic Process Automation in IoT: Enhancing Object Localization Using YOLOv3-Based Class Algorithms.” *International Journal of Information Technology and Computer Engineering* 12 (3): 912–27.

[5] Bobba, Jyothi. 2021. “ENTERPRISE FINANCIAL DATA SHARING AND SECURITY IN HYBRID CLOUD ENVIRONMENTS: AN INFORMATION FUSION APPROACH FOR BANKING SECTORS” 11 (3).

[6] Chetlapalli, Himabindu. 2023. “ENHANCED POST-MARKETING SURVEILLANCE OF AI SOFTWARE AS A MEDICAL DEVICE: COMBINING RISK-BASED METHODS WITH ACTIVE CLINICAL FOLLOW-UP,” June.

[7] Deevi, Durga Praveen. 2022. “Continuous Resilience Testing in AWS Environments with Advanced Fault Injection Techniques” 10 (3).

[8] Dondapati, Koteswararao. 2020. “Robust Software Testing for Distributed Systems Using Cloud Infrastructure, Automated Fault Injection, and XML Scenarios” 8 (2).

[9] Ganesan, Thirusubramanian, Ramy Riad Al-Fatlawy, Suma Srinath, Srinivas Aluvala, and R. Lakshmana Kumar. 2024. “Dynamic Resource Allocation-Enabled Distributed Learning as a Service for Vehicular Networks.” In *2024 Second International Conference on Data Science and Information System (ICDSIS)*, 1–4. https://doi.org/10.1109/ICDSIS61070.2024.10594602.

[10] Gattupalli, Kalyan. 2022. “A Survey on Cloud Adoption for Software Testing: Integrating Empirical Data with Fuzzy Multicriteria Decision-Making” 10 (4).

[11] Gollavilli, Venkata Surya Bhavana Harish, Kalyan Gattupalli, Harikumar Nagarajan, Poovendran Alagarsundaram, and Surendar Rama Sitaraman. 2023. “Innovative Cloud Computing Strategies for Automotive Supply Chain Data Security and Business Intelligence.” *International Journal of Information Technology and Computer Engineering* 11 (4): 259–82.

[12] Gudivaka, Basava Ramanjaneyulu. 2021. “Designing AI-Assisted Music Teaching with Big Data Analysis.” *Current Science*.

[13] Gudivaka, Basava Ramanjaneyulu. 2022. “Real-Time Big Data Processing and Accurate Production Analysis in Smart Job Shops Using LSTM/GRU and RPA.” *International Journal of Information Technology and Computer Engineering* 10 (3): 63–79.

[14] Gudivaka, Rajya Lakshmi, Haider Alabdeli, V Sunil Kumar, C. Sushama, and BalaAnand Muthu. 2024. “IoT - Based Weighted K-Means Clustering with Decision Tree for Sedentary Behavior Analysis in Smart Healthcare Industry.” In *2024 Second International Conference on Data Science and Information System (ICDSIS)*, 1–5. https://doi.org/10.1109/ICDSIS61070.2024.10594075.

[15] Jadon, Rahul. 2018. “Optimized Machine Learning Pipelines: Leveraging RFE, ELM,

and SRC for Advanced Software Development in AI Applications" 6 (1).

[16] Jadon, Rahul. 2019. "Integrating Particle Swarm Optimization and Quadratic Discriminant Analysis in AI-Driven Software Development for Robust Model Optimization" 15 (3).

[17] Jadon, Rahul. 2020. "Improving AI-Driven Software Solutions with Memory-Augmented Neural Networks, Hierarchical Multi-Agent Learning, and Concept Bottleneck Models" 8 (2).

[18] Jadon, Rahul. 2021. "Social Influence-Based Reinforcement Learning, Metaheuristic Optimization, and Neuro-Symbolic Tensor Networks for Adaptive AI in Software Development." *International Journal of Engineering* 11 (4).

[19] Jadon, Rahul, Hitachi Vantara, and Santa Clara. 2019. "Enhancing AI-Driven Software with NOMA, UVFA, and Dynamic Graph Neural Networks for Scalable Decision-Making" 7 (1).

[20] Nagarajan, Harikumar. 2021. "Streamlining Geological Big Data Collection and Processing for Cloud Services" 9 (9726).

[21] "Software Defect Prediction." 2019. 2019. https://www.kaggle.com/datasets/semustafacevik/software-defect-prediction.

[22] Vasamsetty, Chaitanya, and Harleen Kaur. 2021. "OPTIMIZING HEALTHCARE DATA ANALYSIS: A CLOUD COMPUTING APPROACH USING PARTICLE SWARM OPTIMIZATION WITH TIME-VARYING ACCELERATION COEFFICIENTS (PSO-TVAC)." *Journal of Science & Technology (JST)* 6 (5): 132–46.