International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online) Volume 15, Issue 2, February 2026)

Deep Learning Techniques for Software Defect Prediction: A Review

[1]Kapil Kumar, [2]Prof. (Dr.) Vishal Kohli
[1]Research Scholar, Department of Computer Science & Engineering, Neelkanth Institute of Technology, Meerut, India
[2]Director, Department of Computer Science & Engineering, Neelkanth Institute of Technology, Meerut, India

***Abstract*— This review paper presents a comprehensive analysis of deep learning techniques used for software defect prediction. Software defect prediction plays a crucial role in improving software quality, reducing maintenance cost, and enhancing project reliability by identifying error-prone modules at early development stages. The study examines various deep learning models such as Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and hybrid architectures applied to different software datasets. It discusses input features including code metrics, process metrics, and textual features extracted from source code and bug reports. The review also compares performance measures such as accuracy, precision, recall, F1-score, and AUC to evaluate model effectiveness. Furthermore, challenges such as data imbalance, overfitting, feature selection, and model interpretability are highlighted. The paper concludes that deep learning techniques significantly enhance prediction performance compared to traditional machine learning approaches, while emphasizing the need for optimized architectures and explainable models for practical software engineering applications.**

***Keywords*—Software Defects Prediction, CNN, Deep Learning, Performance.**

I. INTRODUCTION

Software defect prediction is an important research area in software engineering that focuses on identifying defective or error-prone modules in a software system before the product is released. In modern software development, systems are becoming more complex due to large codebases, distributed architectures, cloud platforms, and continuous integration practices. As the size and complexity of software increase, the probability of defects also increases[1]. These defects can lead to system failures, security vulnerabilities, financial loss, and reduced user trust. Therefore, predicting defects at an early stage of development is essential to ensure high software quality and reliability[2].

Traditionally, software quality assurance relied on manual testing, code reviews, and static analysis tools. Although these approaches are useful, they are time-consuming and require significant human effort. Moreover, it is difficult to test every part of a large software system thoroughly. Software defect prediction provides a data-driven solution by analyzing historical project data and identifying patterns associated with defective modules[3]. By using machine learning and statistical techniques, defect prediction models can classify software components as defective or non-defective, enabling developers to focus their testing efforts on high-risk areas[4].

Software defect prediction models generally use different types of metrics as input features. These include product metrics such as Lines of Code (LOC), complexity measures (e.g., cyclomatic complexity), coupling, cohesion, and inheritance depth. Process metrics such as number of code changes, developer activity, commit frequency, and bug history are also commonly used[5]. In recent years, textual features extracted from source code, commit messages, and bug reports have been incorporated using Natural Language Processing (NLP) techniques. These diverse features help models capture structural, behavioral, and semantic characteristics of software systems[6].

In early research, traditional machine learning algorithms such as Decision Trees, Naïve Bayes, Logistic Regression, k-Nearest Neighbors, and Support Vector Machines were widely applied for defect prediction. These methods showed promising results but often required manual feature engineering and careful parameter tuning. With the advancement of Artificial Intelligence, deep learning techniques have gained significant attention in this domain[7]. Deep learning models, such as Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM) networks, can automatically learn complex patterns from large datasets without extensive manual feature extraction[8].

Deep learning-based defect prediction models have the ability to capture nonlinear relationships between software metrics and defect occurrence. For example, CNN models can analyze source code as structured data and detect hidden patterns, while RNN and LSTM models can process sequential data such as code changes and commit histories.

These advanced models improve prediction accuracy and reduce false alarms compared to traditional techniques. However, challenges such as data imbalance, limited labeled datasets, overfitting, and lack of interpretability still exist[9].

Another important aspect of software defect prediction is cross-project prediction, where models trained on one project are applied to another. This approach is useful when new projects have limited historical data. Researchers are also exploring transfer learning, ensemble learning, and hybrid machine learning–deep learning approaches to enhance prediction performance. In addition, the integration of defect prediction models into DevOps pipelines and continuous integration systems has become an emerging trend, enabling real-time quality monitoring [10].

II. LITERATURE SURVEY

J. Mythili et al., [1] presented a Software Defect Prediction System using an optimized Bi-Directional LSTM model. The framework captured both forward and backward dependencies in software metrics and historical defect datasets. Feature normalization and hyperparameter tuning were applied to enhance learning performance. The Bi-LSTM model effectively analyzed sequential development data and change logs. Experimental results showed an accuracy of 97.8%, precision of 96.9%, and recall of 98.2%. The model reduced false alarm rate by nearly 4% compared to standard LSTM. The study demonstrated that bidirectional architectures significantly improve defect classification performance.

I. Mehmood et al., [2] presented a machine learning-based framework to improve software defect prediction accuracy. The study incorporated feature selection, ensemble classifiers, and imbalance handling techniques. Multiple algorithms such as Random Forest, SVM, and Gradient Boosting were evaluated. The experimental findings reported an accuracy of 95.6% and F1-score of 94.8%. The ensemble strategy improved AUC value to 0.96 on benchmark datasets. The model also reduced misclassification of minority defect classes. The research highlighted the importance of optimized learning pipelines for reliable defect detection.

S. Muthurajkumar et al., [3] presented a hybrid SwinCNN architecture integrating transformer blocks with convolutional layers. Although designed for agricultural disease prediction, the architecture demonstrated strong feature extraction capability. The hybrid model effectively captured global attention and local spatial features. Performance evaluation achieved an accuracy of 98.1% and F1-score of 97.4%. The transformer-based enhancement improved contextual understanding by approximately 3%. The results showed reduced overfitting compared to standalone CNN models. The architecture indicates strong potential for complex prediction tasks such as software defect analysis.

P. Kaladevi et al., [4] presented an LSTM-based model for analyzing time-series Twitter data to predict crime patterns. The study used NLP preprocessing and sequential modeling techniques. The LSTM effectively captured temporal dependencies in textual datasets. Experimental outcomes reported 93.7% accuracy and 92.5% recall. The model outperformed traditional classifiers by nearly 5% in prediction performance. Results confirmed the capability of recurrent models in sequential data forecasting. Such sequence modeling techniques are applicable to defect prediction using software revision histories.

R. Kabila et al., [5] presented a hybrid LSTM-RNN framework integrated with the Lion Optimization Algorithm for IoT healthcare data management. Optimization enhanced weight adjustment and convergence speed. The hybrid system achieved 96.4% classification accuracy and improved stability during training. The optimization reduced computational complexity by approximately 6%. Experimental validation demonstrated better performance than standalone RNN models. The integration of optimization algorithms strengthened predictive reliability. This approach can enhance software defect prediction models dealing with large datasets.

D. D. Muthusankar et al., [6] presented a Bidirectional Recurrent Neural Network (BiDRNN) model for sentiment analysis. The bidirectional structure processed textual information from both directions to enhance context learning. The model achieved 94.2% accuracy and an F1-score of 93.5%. Dropout layers were used to minimize overfitting and improve generalization. Comparative analysis showed nearly 4% improvement over conventional RNN. The study validated the effectiveness of bidirectional deep learning architectures. Similar sequential modeling strategies can be applied in defect prediction tasks.

K. D. Kumar et al., [7] presented a deep CNN framework combined with handcrafted features for sign language recognition. The integration of learned and engineered

features improved classification robustness. The CNN effectively extracted spatial representations from image inputs. Experimental evaluation showed 97.3% accuracy and improved precision values. The hybrid feature approach enhanced recognition rate by approximately 3% compared to baseline CNN. The system also demonstrated strong generalization capability. This combination strategy can be extended to software defect prediction using metric-based and semantic features.

A. Gnanabaskaran et al., [8] presented a hybrid prediction model using Graph Neural Networks (GNN) and Support Vector Machine (SVM) for drug-drug interaction analysis. The GNN captured relational dependencies in graph-structured data. Feature embeddings generated by GNN were classified using SVM for improved precision. Experimental results reported 95.1% accuracy and 0.95 AUC value. The hybrid model reduced classification error by nearly 4%. The approach demonstrated efficient learning from structured datasets. Such graph-based techniques are applicable to dependency-based defect prediction models.

M. Azzeh et al., [9] presented an analysis of kernel-based Support Vector Machine methods for software defect prediction. Different kernel functions such as linear, polynomial, and RBF were evaluated. The RBF kernel achieved the highest performance with 92.8% accuracy. The study reported an AUC value of 0.93 and F-measure of 91.6%. Results highlighted the importance of selecting appropriate kernel functions. Comparative experiments showed improved stability across datasets. The research provides a strong benchmark for evaluating deep learning-based defect prediction models.

R. Praveen et al., [10] presented a fusion model combining Convolutional Neural Networks and Capsule Networks for classification tasks. The architecture preserved hierarchical relationships through capsule layers. The fusion strategy improved feature representation and minimized information loss. Experimental results demonstrated 96.7% accuracy and enhanced recall performance. The model outperformed standalone CNN by nearly 4%. Capsule integration strengthened pattern recognition capability. Such fusion-based deep learning models can enhance software defect prediction accuracy and robustness.

III. CHALLENGES

Software defect prediction has achieved significant progress with machine learning and deep learning techniques, but several challenges still limit its practical effectiveness. The quality and availability of historical project data greatly influence prediction performance. Many datasets are noisy, incomplete, or inconsistent, which reduces model reliability. In addition, defect datasets are often imbalanced, where defective modules are much fewer than non-defective ones. Deep learning models also require large amounts of labeled data, which is not always available in real-world projects. Furthermore, interpretability, scalability, and cross-project generalization remain open research issues. These challenges must be carefully addressed to build robust and industry-ready defect prediction systems.

1. Data Imbalance Problem

Most software defect datasets contain significantly fewer defective modules compared to non-defective ones. This imbalance causes models to favor the majority class, leading to poor recall for defect detection. As a result, many actual defective modules may remain undetected. Techniques such as oversampling, undersampling, and synthetic data generation are required to handle this issue effectively.

2. Limited Labeled Data

Deep learning models require large volumes of labeled training data to perform well. However, many software projects do not maintain detailed defect history. New projects especially lack sufficient historical records. This limitation reduces the ability of complex models to generalize accurately.

3. Feature Selection and Engineering

Selecting relevant software metrics is a difficult task. Irrelevant or redundant features can reduce prediction accuracy and increase computational cost. Manual feature engineering requires domain expertise. Although deep learning can automatically extract features, improper feature representation still affects performance.

4. Overfitting in Deep Models

Deep neural networks have large numbers of parameters, which increases the risk of overfitting. When trained on small or noisy datasets, models may memorize patterns instead of learning general relationships. This reduces prediction accuracy on unseen data. Regularization and dropout techniques are often required to minimize this issue.

5. Cross-Project Prediction Issues

Models trained on one project often perform poorly when applied to another project. Differences in coding standards, programming languages, team practices, and project size create distribution mismatch. This limits the practical usability of defect prediction systems across organizations.

6. Lack of Interpretability

Many deep learning models act as black-box systems. Developers may find it difficult to understand why a module is predicted as defective. Lack of explanation reduces trust and acceptance of automated systems in software engineering environments.

7. Scalability and Computational Cost

Large-scale software systems generate massive amounts of data. Training deep learning models on such datasets requires high computational power and memory resources. This increases infrastructure cost and limits adoption in smaller organizations.

8. Dynamic and Evolving Software Systems

Software systems continuously evolve through updates and new features. Models trained on old data may become outdated due to concept drift. Continuous retraining and model adaptation are necessary to maintain prediction accuracy over time.

These challenges through advanced data preprocessing, transfer learning, explainable AI, and adaptive learning strategies will significantly enhance the effectiveness of software defect prediction systems.

IV. CONCLUSION

Software defect prediction plays a vital role in improving software quality, reliability, and maintenance efficiency by identifying error-prone modules at early development stages. The integration of machine learning and deep learning techniques has significantly enhanced prediction accuracy, recall, and overall performance compared to traditional statistical approaches. Advanced models such as CNN, LSTM, hybrid architectures, and ensemble frameworks demonstrate strong capability in handling complex software metrics and sequential development data. However, challenges including data imbalance, limited labeled datasets, overfitting, cross-project generalization, and lack of interpretability still restrict full industrial adoption. Future research should focus on explainable AI models, transfer learning, automated feature extraction, and scalable architectures to build more reliable and practical defect prediction systems for real-world software engineering environments.

REFERENCES

1. J. Mythili, T. Suganraj, V. Suvetha, M. Thamaraikannan and S. M. Polisetty, "Software Defect Prediction System Using Optimized Bi-Directional LSTM," *2025 International Conference on Advanced Computing Technologies (ICoACT)*, Sivalasi, India, 2025, pp. 01-06, doi: 10.1109/ICoACT63339.2025.11004992.
2. I. Mehmood, S. Shahid, H. Hussain, I. Khan, S. Ahmad, S. Rahman, N. Ullah, and S. Huda, “A novel approach to improve software defect prediction accuracy using machine learning,” IEEE Access, vol. 11, pp. 63 579–63 597, 2023.
3. S. Muthurajkumar and G. K. Kumar, “Swincnn: A hybrid deep learning architecture for accurate cotton disease prediction,” in 2023 12th International Conference on Advanced Computing (ICoAC). IEEE, 2023, pp. 1–7.
4. P. Kaladevi, M. Gopalraj, K. Harish, A. Guna, and R. Praveen, “Tweets to predict: Lstm model for crime analysis using twitter time series data,” in 2024 International Conference on Knowledge Engineering and Communication Systems (ICKECS), vol. 1. IEEE, 2024, pp. 1–6.
5. R. Kabila, S. S. Balaji, V. Vikraam, S. R. Kumar, and R. Praveen, “Hybrid lstm-rnn and lion optimization algorithm for iot-based proactive healthcare data management,” in 2024 International Conference on Knowledge

Engineering and Communication Systems (ICKECS), vol. 1. IEEE, 2024, pp. 1–7.

6. D. D. Muthusankar, D. P. Kaladevi, D. V. Sadasivam, and R. Praveen, "Bidrn: A method of bidirectional recurrent neural network for sentiment analysis," arXiv preprint arXiv: 2311.07296, 2023.
7. K. D. Kumar, K. Ragul, G. P. P. Kumar, and G. K. Kumar, "Enhancing sign language recognition through deep cnn and handcrafted features," in 2024 2nd International Conference on Networking and Communications (ICNWC). IEEE, 2024, pp. 1–6.
8. A. Gnanabaskaran, K. T. Bharathi, S. Nandakumar, B. Sanjay, and R. Praveen, "Enhanced drug-drug interaction prediction with graph neu-ral networks and svm," in 2024 International Conference on Knowledge Engineering and Communication Systems (ICKECS), vol. 1. IEEE, 2024, pp. 1–6.
9. M. Azzeh, Y. Elsheikh, A. B. Nassif, and L. Angelis, "Examining the performance of kernel methods for software defect prediction based on support vector machine," Science of Computer Programming, vol. 226, p. 102916, 2023.
10. R. Praveen, P. Pabitha, V. Sakthi, S. Madhavi, and R. V. Sai, "Convolutional neural network and capsule network fusion for effective attrition classification," in 2023 12th International Conference on Advanced Computing (ICoAC). IEEE, 2023, pp. 1–6.