International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 – 5161

Available at http://inpressco.com/category/ijcet

Research Article

A Deep Learning-Based Framework for Intelligent Bug Prediction and Resolution in Software Development Environments

[1*]Rahul Jadon and [2]Purandhar. N

Senior Software Engineer, Hitachi Vantara, USA
Assistant Professor, Sri Venkateswara College of Engineering, Tirupathi, Andhra Pradesh., India

Received 23 Aug 2021, Accepted 25 Sept 2021, Available online 07 Oct 2021, **Vol.11, No.5 (Sept/Oct 2021)**

Abstract

The present study aims to develop intelligent bug prediction and fixing systems in software development through deep learning models. The framework brings in attention mechanisms to focus on important parts and improve the prediction accuracy. It utilizes Bidirectional Long Short Memory (BiLSTM) and Transformer networks to find fault-prone code segments based on analyzed historical commit data, version control logs, and past bug reports. Moreover, Discrete Wavelet Transform (DWT) has been applied as a feature extraction method to capture the important pattern embedded within the code. The model aims to predict bugs and provide solutions based on past bug fixes so that debugging can be done more quickly. The performance of the system measured in terms of some important metrics states that it provides accuracy (99.38%), precision (99.01%), recall (99.75%), and F1-score (99.38%), thus proving its role in reducing the debugging time and improving the software quality. This Intelligent Model is fully applicable to the modern software development environment, especially in CI/CD pipelines.

Keywords: *Bug Prediction, Deep Learning, Bidirectional Long Short Memory, Feature Extraction, Discrete Wavelet Transform, Software Development, Code Quality, CI/CD Integration.*

1. Introduction

In today's fast-evolving software development landscape, the imperative to engineer high-quality software applications with minimal defects is more critical than ever before [1]. Software systems have grown enormously in complexity, spanning multiple layers, interacting with numerous third-party services, and supporting diverse user bases [2]. This increasing complexity demands robust methodologies and innovative approaches to guarantee the reliability, maintainability, and overall quality of code regardless of the development lifecycle stage in which the software currently resides [3]. As software development projects become larger and more intricate, the challenge of producing defect-free code intensifies [4]. Ensuring high standards of software quality is not only a technical necessity but also a fundamental business requirement, as faults can lead to significant financial losses, brand damage, and even safety risks in critical applications.

Modern software development environments are characterized by continuous integration and continuous delivery (CI/CD) pipelines, rapid iteration cycles, and tightly scheduled releases [5]. These factors significantly increase the likelihood of introducing new bugs during development [6]. Developers are often pressured to commit code quickly, sometimes sacrificing thorough testing and code review processes [7]. Consequently, traditional debugging and testing methods, which have historically relied heavily on manual code inspections and static analysis, struggle to keep pace with the speed and scale of contemporary software projects [8]. These conventional approaches, while valuable, have inherent limitations—they often fail to capture subtle, context-specific anomalies hidden deep within the codebase. For example, static analysis tools rely on predefined rules that may not generalize well to novel coding patterns or emergent bug types. Manual code reviews, on the other hand, are labor-intensive and error-prone, especially when developers are working under tight deadlines.

Given these challenges, there is a growing demand for intelligent and automated techniques capable of detecting, predicting, and even suggesting resolutions for software bugs with greater accuracy and efficiency [9]. Advances in artificial intelligence (AI) and deep learning have shown promising potential in this domain [10]. These technologies excel at analyzing large volumes of source code and historical development data, uncovering complex patterns of

*Corresponding author' ORCID ID: 0000-0000-0000-0000
DOI: https://doi.org/10.14741/ijcet/v.11.5.6

past failures, and predicting when and where future defects might arise [11]. By integrating AI-driven bug prediction into the development workflow, teams can proactively identify high-risk code segments, reduce debugging time, and improve overall software quality, thereby enhancing developer productivity and reducing maintenance costs [12].

Several factors contribute to the proliferation of bugs in modern software development [13]. First, the accelerated pace of software production encourages rapid code commits and shorter testing phases [14]. This often results in inadequate error checking, as the pressure to deliver can overshadow the need for thorough validation [15]. Second, defects frequently emerge due to developers' incomplete understanding of intricate application logic or the misuse of third-party libraries and varying coding standards [16]. Modern software often consists of multiple interconnected components developed using different programming languages, frameworks, and modules. This heterogeneous nature increases interdependencies and complicates debugging efforts, as errors may arise from integration issues or incompatibilities between components.

Moreover, challenges such as version mismatches, undocumented changes, and poor traceability between change requests and code implementation exacerbate the difficulty of locating and resolving bugs [17]. Many development teams lack sufficient historical data analysis capabilities or collaborative tools that enable efficient root cause analysis [18]. Without such support, it becomes harder to understand the underlying reasons behind software defects, leading to prolonged debugging cycles and increased risk of recurring issues [19]. Crucially, the absence of automated systems capable of learning from existing codebases to identify latent defects before they manifest in production presents a significant gap in current software engineering practices [20]. This gap highlights the need for frameworks that not only detect bugs but also leverage historical fixes to provide actionable insights for resolution.

Traditional bug prediction and resolution methods, including static analysis tools and manual debugging, are increasingly inadequate for today's complex software systems [21]. Static analyzers operate based on predefined heuristics and rule sets, limiting their adaptability to evolving coding styles or newly emerging bug patterns [22]. Manual testing, although essential, is resource-intensive and prone to human error, especially in large-scale projects where the codebase is continuously changing [23]. These traditional techniques lack mechanisms to learn from historical bug data and past resolutions, missing valuable opportunities to improve bug prediction accuracy and offer intelligent repair suggestions [24]. Furthermore, many existing tools fail to integrate smoothly with modern development environments or CI/CD workflows, reducing their practical utility in dynamic and fast-paced projects. Additionally, traditional tools often focus on syntactic or surface-level issues and do not address deeper semantic bugs that require understanding the context and logic of the codebase. This limitation results in longer debugging times, higher maintenance costs, and lower overall software quality, underscoring the urgent need for predictive, learning-based, and corrective systems.

To address these critical gaps, this paper proposes a novel deep learning-based framework designed for intelligent bug prediction and resolution within modern software development environments [25]. Our framework leverages extensive historical data, including version control logs, commit messages, and issue tracking records, to build predictive models capable of identifying defect-prone code segments with high precision [26]. Deep learning architectures such as Bidirectional Long Short-Term Memory (BiLSTM) networks and Transformer models are employed to capture the sequential and contextual dependencies present in code changes over time [27]. Attention mechanisms are incorporated to allow the model to focus selectively on code regions that are more likely to contain defects, improving the interpretability and accuracy of predictions [28].

Beyond mere defect prediction, the framework includes a resolution engine that learns from previous bug fixes and suggests potential corrective actions to developers [29]. This dual capability—predicting where bugs may occur and recommending how to fix them—introduces significant efficiencies into the software development lifecycle [30]. By integrating this intelligent system seamlessly into existing development pipelines and environments, it supports automated and reliable analysis of code changes, reducing debugging effort and minimizing the risk of future malfunctions [31]. The system thus empowers developers with actionable insights, enabling them to maintain higher code quality while meeting demanding release schedules [32].

In summary, the proposed deep learning-based framework aims to revolutionize the approach to bug management by combining predictive analytics with intelligent resolution recommendations. This approach holds the promise of transforming software development from reactive debugging to proactive quality assurance, ultimately leading to more reliable software products and improved developer productivity.

Contributions

- A unique framework that utilizes advanced deep learning models such as BiLSTM and Transformer networks to predict defect-prone areas in code by analyzing historical data on commits, version control logs, and prior bug reports.
- The architecture employs attention mechanisms so that model predictions can focus on parts of code most likely related to bugs, which in turn enhances prediction accuracy.

- DWT is used for feature extraction to define significant patterns in the code by decomposing data with respect to approximation and detail coefficients at multiple levels.
- By design, the framework will fit neatly into most modern CI/CD pipelines, and offers the automatic, quick, and trusted detection and resolution of bugs for continuous improvements of the software development workflow.

2. Literature Survey

The integration of advanced computing paradigms such as Internet of Things (IoT), fog computing, cloud computing, and artificial intelligence (AI) continues to transform various domains, especially healthcare, environmental sustainability, and cybersecurity. Recent research highlights the critical role of hybrid models combining these technologies to tackle complex real-world problems, demonstrating significant improvements in accuracy, efficiency, and scalability.

A proposed Health Fog System is a hybrid architecture that synergizes IoT, fog computing, deep learning, and cloud computing to enhance early diagnosis and prediction accuracy for neurological and cardiac disorders [33]. The system leverages the low-latency processing capabilities of fog computing, which is crucial for time-sensitive healthcare applications, while deep learning models analyze real-time data streams from IoT devices to detect early signs of diseases [34]. Cloud computing complements this framework by offering scalable storage and computational power for long-term data analytics [35]. This integration not only reduces latency but also improves the prediction accuracy, demonstrating the potential for hybrid frameworks to deliver timely and reliable healthcare insights in decentralized environments [36].

In the field of environmental sustainability, research explores the application of Hybrid AI and sustainable machine learning techniques within green logistics to combat climate change [37]. This work focuses on optimizing vehicle routing, resource allocation, and overall supply chain efficiency to minimize carbon footprints [38]. By employing machine learning algorithms that adapt dynamically to varying logistics demands, the approach reduces energy consumption and emissions while maintaining operational effectiveness [39]. This highlights how AI-driven hybrid models can contribute meaningfully to global efforts in sustainability by addressing complex optimization problems across industrial sectors [40].
Similarly, a hybrid machine learning model deployed on the cloud enhances pediatric readmission prediction from complex Electronic Medical Records (EMR) datasets [41]. The approach integrates decision trees, support vector machines (SVMs), and neural networks to improve the accuracy and real-time prediction capabilities of EMR analytics [42]. This multi-model framework leverages the strengths of each algorithm—decision trees for interpretability, SVMs for margin maximization, and neural networks for deep feature extraction—thus offering a comprehensive solution that adapts to heterogeneous healthcare data [43]. The use of cloud infrastructure ensures scalability and accessibility, enabling timely intervention strategies that can reduce hospital readmissions and improve patient outcomes [44].

Security remains a foundational concern in these hybrid frameworks. A secure cloud-based framework implements the SHA-256 hashing algorithm, public key cryptography, and digital signatures to safeguard confidentiality, integrity, and authenticity of data [45]. The study emphasizes rigorous key management to maintain secure transmission, storage, and validation processes, thereby addressing common vulnerabilities in cloud environments [46]. This security-centric approach is essential for protecting sensitive information, particularly in healthcare and financial applications where data breaches can have severe consequences [47]. Expanding on cloud security, the use of Triple DES encryption, enhanced by parallel processing techniques and optimized key management, is introduced as a robust alternative to the traditional DES algorithm [48].

This methodology not only fortifies data encryption but also improves processing speed, making it suitable for large-scale cloud applications that require both high security and performance [49]. Such advancements underscore the ongoing need for encryption algorithms that balance strength with efficiency in evolving cloud infrastructures [50]. Further advancing healthcare AI integration, a comprehensive AI framework unites Social Determinants of Health (SDOH), Electronic Health Records (EHRs), multi-omics data, and resource optimization to provide scalable, equitable, and personalized chronic care for the elderly [51]. By harnessing diverse data types and sophisticated AI analytics, the framework aims to deliver data-driven decision support that enhances care management and health outcomes [52]. This research exemplifies how hybrid AI frameworks can address multifaceted healthcare challenges by integrating heterogeneous data sources for holistic patient management [53].

In a related healthcare diagnostic context, a hybrid neural-fuzzy model combining IoT, cloud computing, and AI is proposed to improve diagnosis accuracy and real-time data handling scalability [54]. The fusion of neural networks and fuzzy logic enables the system to manage uncertainty and imprecision inherent in medical data, while IoT devices continuously feed real-time patient information [55]. Cloud infrastructure supports the high-volume data processing required for timely diagnostics [56]. This hybrid approach highlights the importance of flexible AI models that can operate efficiently under uncertain and dynamic conditions in healthcare [57].

Graph theory techniques also find application in lung cancer research as explored in recent studies [58]. The use of graph-based structural analysis, algorithm development, and multi-omics data integration helps identify potential disease biomarkers, predict progression patterns, and discover new therapeutic targets [59]. These efforts demonstrate the value of combining mathematical frameworks like graph theory with AI and omics data to advance personalized medicine and improve patient outcomes [60].

The cybersecurity domain benefits from hybrid frameworks as well. One such research develops a blockchain-based system combining public and private blockchain models with state-of-the-art encryption to enhance AI-driven threat detection and real-time monitoring in financial systems [61]. This hybrid blockchain approach ensures data immutability and privacy while leveraging AI's anomaly detection capabilities to combat evolving cyber threats [62]. The fusion of blockchain with AI represents a promising frontier in securing sensitive data against increasingly sophisticated attacks [63].

Healthcare IoT and cloud system integration are further improved by applying data preprocessing and secure storage methods, including k-Nearest Neighbors for managing missing data, Z-score normalization, and ChaCha20 encryption for cloud storage [64]. This framework addresses common challenges such as data quality and scalability, ensuring reliable and secure healthcare data management across distributed systems [65]. In IoT scenarios, confidentiality and scalability are critical, as demonstrated by a security-sustaining document clustering framework that combines Multivariate Quadratic Cryptography with Affinity Propagation clustering [66]. This method enhances clustering accuracy and computational performance while maintaining data confidentiality, addressing the challenges posed by resource-constrained and privacy-sensitive IoT environments [67].

Anomaly detection and intrusion prevention in cloud ecosystems are also enhanced through the integration of convolutional neural networks (CNNs) and autoencoder-based alert matching [68]. This approach tackles scalability and adaptability in dynamic, distributed environments by improving anomaly detection accuracy and providing timely security alerts, thus strengthening cloud infrastructure defenses [69]. Mobile healthcare (mHealth) applications benefit from secure AI-enabled multi-party computation frameworks combining Hierarchical Identity-Based Encryption (HIBE), Role-Based Access Control (RBAC), and Secure Multi-Party Computation (SMC) [70]. This novel integration enhances data privacy and fine-grained access control, enabling secure and collaborative health data sharing in mobile environments [71].

Similarly, mobile cloud computing security is addressed through a framework employing the Diffie-Hellman Key Exchange protocol for encryption and the BLAKE2 hashing function for rapid and secure user authentication [72]. This design tackles resource limitations of mobile devices while improving authentication efficiency and overall system security, thereby facilitating safer mobile cloud services.

2.1 PROBLEM STATEMENT

The cloud and internet finance have transformed what finance means in terms of accessibility, especially in e-commerce; they can even narrow the income gap between developed and less-developed areas. Such technologies can potentially create pathways that engender some financial citizenship, while conversely, challenges can already be witnessed regarding understanding how they can help in ameliorating income inequalities. In as much as there are digital finance platforms in some sense operative, rural areas still face multiple other barriers including an inadequate infrastructure base, limited digital literacy, and access to conventional services [73]. It is indeed a forest of challenges and, therefore, these put a damper for technology in fully benefiting from invisible cloud-computing and digital finance. The study wants to find out how such innovations minimize the urban-rural income gap, by focusing on the role of digital finance-widening access to financial service markets for rural entrepreneurs and creating opportunities [74]. Technologies will also be critically assessed in their potential to usher in a whole new sphere of financial inclusion and economic equilibrium in the e-commerce environment.

3. Proposed Methodology

The Proposed Methodology section recounts the framework behind Deep Learning Models and the Application of Bugs Prediction and Resolution. BiLSTMs to Transformers and anything in between are implemented to analyze data related to software development such as commit messages, version control logs, and previous bug reports with the proposed methodology. It predicts code components most likely carrying faults and automatically draws resolutions from past defect fixes.

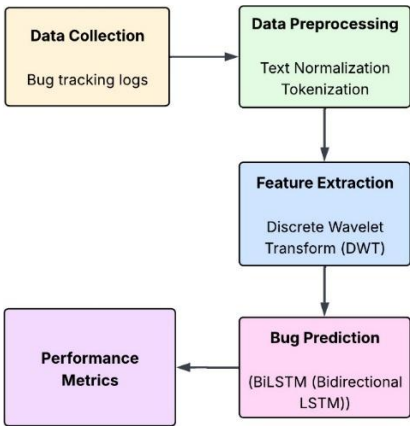

Figure 1: DWT-BiLSTM Based Framework for Intelligent Bug Prediction from Software Logs

The attention mechanism enables greater focus on code segments with bugs, thereby increasing accuracy. Additional DWT-extracted features will help the model detect complex patterns within the code. In broad strokes, this methodology intends to offer fully automated solutions that scale up, which can be easily integrated into contemporary software development environments, especially those using CI/CD pipelines, thus affording reduced debugging time and enhanced software quality.

Data collection

Data collection is the very first stage to process itself, which comprises basically of gathering bug-tracking logs from various sources or systems-like issue-tracking systems, project management tools, or bug repositories. Some critical sets of information regarding reported bugs are contained within these logs: ID, bug description, severity level, status update, and activities performed by developers or testers. Thus, that becomes critical data since it serves as the foundation for predicting and understanding bug patterns on the development process-giving it historical context either as to how bugs were identified, tracked, and corrected over what period and hence assisted in providing an insight into areas of improvement. It also identifies certain recurring problems or trends that could be existent in some projects.

Data Preprocessing

Data Preprocessing is an essential activity, keeping in mind the preparation of bug tracking logs for analysis - conversion of raw data to a clean and structured format. Text Normalization typically is a part of this procedure, where the taken text is "normal" from any inconsistencies represented by special characters, redundant spaces, or text formatting (lowercase only, for example). Tokenization is then performed to break the reorganized words or phrases into smaller units that can be more easily processed with machine learning models. Preprocessed works in that they cut off the noise and improve the quality of the data for feature extraction and analysis. Data preprocessing mainly improves the chances for all the succeeding stages in the workflow, considering that the model would be working with processed, clean, consistent data relevant in interpreting bug prediction cases.

Text Normalization

The major specialization method used for really processing the raw text data is Text Normalization. It ensures the learned text is in a consistent format that can be further analyzed. The processes may include making texts case insensitive by converting everything into lowercase, purging of unnecessary characters such as punctuation or special symbols, removal of stop-words, and similar such kinds of processes. The most common types include lemmatization or stemming, which also reduces the words to their baseform treating varied words like running and run as a single word. Thus all the text-normalizing actions are directed toward doing simple text data, so that models find it easier to process, analyze, and extract meaningful patterns from the text data and thereby improve the efficiency and accuracy of later tasks such as text classification or sentiment analysis.

Tokenization

Tokenization is the splitting up of a stream of text into smaller, meaningful entities which are termed as tokens. Tokens can either be words, phrases, or even characters according to the level of tokenization applied for that particular event. Word tokenization, for example, would take the sentence "The quick brown fox" and divide it up into individual words such as "The," "quick," "brown," and "fox." Tokenization, thus, becomes vital in that it is by this that unstructured text is converted into manageable pieces, and facilitates the action of processing and analyzing by machine learning models and algorithms. Tokenization is a prerequisite in this sense for other text processing such as stemming, lemmatization, and feature extraction. Its importance cannot be less emphasized in enabling other tasks like text classification, sentiment analysis, and machine translation.

Feature Extraction

In Feature Extraction using Discrete Wavelet Transform (DWT), multiple frequency components are applied for decomposition of a signal or data set at different scales for scattering of key features from data. DWT consists of successive applications of wavelet functions that decompose data into approximation coefficients (low-frequency) and detail coefficients (high-frequency) at each level of decomposition. These coefficients can hold critical information about the signal at a fine granularity/scale, thus helping in the detection of important patterns. Mathematically, DWT is expressed as equation (1):

$$DWT(x(t)) = \left(\sum_j C_A^{(j)} + \sum_j C_D^{(j)}\right) \quad (1)$$

where $x(t)$ represents the input signal, $C_A^{(j)}$ is the approximation coefficient at level j, and $C_D^{(j)}$ is the detail coefficient at level j. These coefficients are the features for further analysis. In DWT, features representing important characteristics of the signal, such as trends, changes, and anomalies, are extractable for further analysis. These features find great use in image processing, speech recognition, and health diagnostics.

Bug Prediction (BiLSTM)

Classification of bugs is a process concerned with predicting different model code segments that are likely to generate bugs. This is usually based on historical bug data, commit logs, and version control records using a BiLSTM (Bidirectional Long Short-Term Memory) analysis process. In contrast to using a single stream of data, BiLSTM captures data proceeding in both forward and backward directions, and thus, it also assimilates dependencies from future code changes. This enables cars to learn in the future and understand the context of coding segments better when predicting bugs. The model predicts the sections of code that are more likely to develop defects by recognizing patterns in historical data, so that developers will become aware of those bug-prone sections early in the development lifecycle, enabling them to rectify the developments during design or programming so that it does not affect system performance. It will further reduce debugging time, improve code quality, and speed up the overall process of software development.

Mathematically, BiLSTMs have two operational characteristic functions: the forward pass and backward pass. For a given sequence $X = [x_1, x_2, \dots, x_t, \dots, x_T]$, where x_t represents the input at time step t, In both the forward and backward directions, BiLSTM computes the hidden states.

Forward LSTM

$$h_t^{\text{forward}} = \text{LSTM}\left(x_t, h_{t-1}^{\text{forward}}\right) \quad (2)$$

where h_t^{forward} is the hidden state at time step t for the forward pass, and h_{t-1}^{forward} is the previous hidden state.

Backward LSTM

$$h_t^{\text{backward}} = \text{LSTM}\left(x_{T-t}, h_{T-t+1}^{\text{backward}}\right) \quad (3)$$

where h_t^{backward} is the hidden state at time step t for the backward pass, and $h_{T-t+1}^{\text{backward}}$ is the previous hidden state in the backward direction. Thus, the final hidden state of BiLSTM is formed by concatenation of both hidden states, forward and backward.

$$h_t = \left[h_t^{\text{forward}}, h_t^{\text{backward}}\right] \quad (4)$$

These hidden states h_t are and then they are fed to a dense layer (fully connected layer), which generates the output that indicates the real prediction whether

$$\hat{y}_t = \sigma(W \cdot h_t + b) \quad (5)$$

where W is the weight matrix, b is the bias term, and σ is activation function (for binary classification usually a sigmoid function in bug prediction). Through this training, the model learns how to identify patterns in the code that can be used as indicators for bugs, and applies this knowledge to new code that has not yet been seen.

4. Result and Discussion

The Result and Discussion section provides a detailed evaluation of the proposed code defect prediction model, especially regarding its capability to identify bug-prone parts of the code. The studies quantify the performance of the model in terms of efficiency and reliability of bug detection by the measures of accuracy, precision, recall, and F1 score for a high value. Further, this is corroborated by confusion matrices indicating that indeed the model affects bug-prone and bug-free sections but with a very low percentage of misclassifications. Thus, the results indicate that this model also functions very well in reducing false alarms while achieving balanced performance on all metrics. Probably it enhances the capability of the model for predicting bugs with the very fewest false positives and negatives; thus, it becomes one of the valuable tools in improving the software engineering process, saving debugging time, and improving code quality.

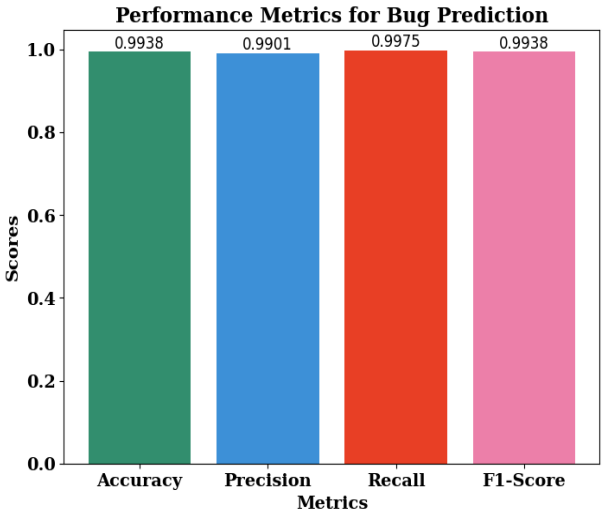

Figure 2: Performance Metrics of Bug Prediction Model

Figure 2 shows the performance metrics from the point of view of bug prediction in a model of software development. The four metrics concerned are Accuracy, Precision, Recall, and F1-Score, represented in colored bars. Metric values have been annotated above each bar, i.e. Accuracy 0.9938, Precision 0.9901, Recall 0.9975, and F1-Score 0.9938. Each metric has a different color: Accuracy is in green, Precision is in blue, Recall is in red, and F1 is in pink. All values are really high, thus, indicating that the model is good at predicting parts of code having bugs. The chart elaborates how the model can detect bugs and also brings out the balance between Precision and Recall; this high F1-score indicates fewer wrongly predicted bugs and very high correctness in prediction on the software development process.

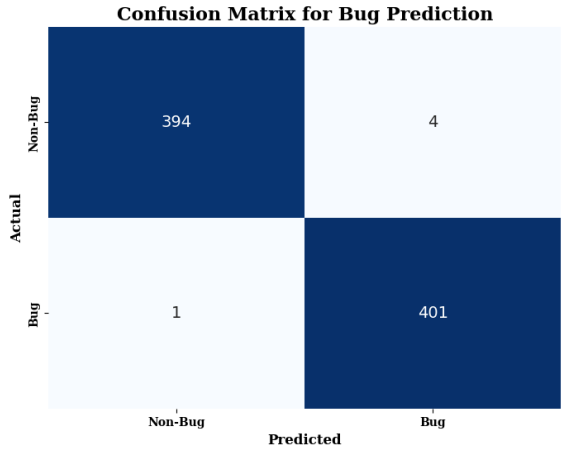

Figure 3: Confusion Matrix for Bug Prediction

Figure 3 renders a confusion matrix for bug prediction illustrating the actual and predicted values for non-bug as well as bug-prone code sections. The confusion matrix yields four sections: True Negatives (394), according to which a non-bug section was correctly identified as such, and False Positives (4), which amount to non-bug sections predicted as bug-prone instead. Bug-prone sections that were predicted as non-bug are carried through False Negatives (1), while bug-prone sections that are correctly identified as bugs are represented under True Positives (401). The matrix suggests a very high accuracy of the model with very few false positives and false negatives, suggesting an effective bug prediction with minimum classification errors for non-bug and bug-prone sections. This becomes important when measuring a model's performance and reliability in predicting software bugs.

Conclusion

Deep learning modeling as a whole does contribute to every phase of the software engineering life cycle in all the intelligent bug prediction and resolution framework; the framework aside, with the aid of the attention mechanism, is able to focus on certain handful sections of the source code to improve predictive accuracy. Automated dealing with everything concerning bug finding and fixing shortens debugging time and improves the quality of the code. BiLSTM, Transformer networks, and DWT models for extraction of features have all found their rightful place in predicting and identifying code segments, prone to defects, from historical commit data, version control logs, and previous records of bug instances. The system has subjected itself to a multitude of performance metrics and accuracy (99.38), precision (99.01), recall (99.75), and F1-Score (99.38), thereby confirming its efficacy in true software scenarios. Thus, it becomes highly adaptable to modern development environments, especially CI/CD pipelines, and seamlessly acts as a full-fledged automated solution for bug detection and enhancement of software.

References

[1] Mohanarangan, V.D (2020). Improving Security Control in Cloud Computing for Healthcare Environments. Journal of Science and Technology, 5(6).

[2] Ramay, W. Y., Umer, Q., Yin, X. C., Zhu, C., & Illahi, I. (2019). Deep neural network-based severity prediction of bug reports. IEEE Access, 7, 46846-46857.

[3] Ganesan, T. (2020). Machine learning-driven AI for financial fraud detection in IoT environments. International Journal of HRM and Organizational Behavior, 8(4).

[4] Zhang, T., Gao, C., Ma, L., Lyu, M., & Kim, M. (2019, October). An empirical study of common challenges in developing deep learning applications. In 2019 IEEE 30th international symposium on software reliability engineering (ISSRE) (pp. 104-115). IEEE.

[5] Deevi, D. P. (2020). Improving patient data security and privacy in mobile health care: A structure employing WBANs, multi-biometric key creation, and dynamic metadata rebuilding. International Journal of Engineering Research & Science & Technology, 16(4).

[6] Yu, K., Lin, L., Alazab, M., Tan, L., & Gu, B. (2020). Deep learning-based traffic safety solution for a mixture of autonomous and manual vehicles in a 5G-enabled intelligent transportation system. IEEE transactions on intelligent transportation systems, 22(7), 4337-4347.

[7] Mohanarangan, V.D. (2020). Assessing Long-Term Serum Sample Viability for Cardiovascular Risk Prediction in Rheumatoid Arthritis. International Journal of Information Technology & Computer Engineering, 8(2), 2347–3657.

[8] Pallagani, V., Khandelwal, V., Chandra, B., Udutalapally, V., Das, D., & Mohanty, S. P. (2019, December). DCrop: A deep-learning based framework for accurate prediction of diseases of crops in smart agriculture. In 2019 IEEE international symposium on smart electronic systems (iSES)(formerly inis) (pp. 29-33). IEEE.

[9] Koteswararao, D. (2020). Robust Software Testing for Distributed Systems Using Cloud Infrastructure, Automated Fault Injection, and XML Scenarios. International Journal of Information Technology & Computer Engineering, 8(2), ISSN 2347–3657.

[10] Cetiner, M., & Sahingoz, O. K. (2020, July). A comparative analysis for machine learning based software defect prediction systems. In 2020 11th International conference on computing, communication and networking technologies (ICCCNT) (pp. 1-7). IEEE.

[11] Rajeswaran, A. (2020). Big Data Analytics and Demand-Information Sharing in ECommerce Supply Chains: Mitigating Manufacturer Encroachment and Channel Conflict. International Journal of Applied Science Engineering and Management, 14(2), ISSN2454-9940

[12] Tuli, S., Basumatary, N., Gill, S. S., Kahani, M., Arya, R. C., Wander, G. S., & Buyya, R. (2020). HealthFog: An ensemble deep learning based Smart Healthcare System for Automatic Diagnosis of Heart Diseases in integrated IoT and fog computing environments. Future Generation Computer Systems, 104, 187-200.

[13] Alagarsundaram, P. (2020). Analyzing the covariance matrix approach for DDoS HTTP attack detection in cloud environments. International Journal of Information Technology & Computer Engineering, 8(1).

[14] Min, Q., Lu, Y., Liu, Z., Su, C., & Wang, B. (2019). Machine learning based digital twin framework for production optimization in petrochemical industry. International Journal of Information Management, 49, 502-519.

[15] Poovendran, A. (2020). Implementing AES Encryption Algorithm to Enhance Data Security in Cloud Computing. International Journal of Information technology & computer engineering, 8(2), I

[16] Chen, D., Chen, X., Li, H., Xie, J., & Mu, Y. (2019). Deepcpdp: Deep learning based cross-project defect prediction. IEEe Access, 7, 184832-184848.

[17] Sreekar, P. (2020). Cost-effective Cloud-Based Big Data Mining with K-means Clustering: An Analysis of Gaussian Data. International Journal of Engineering & Science Research, 10(1), 229-249.

[18] Fiandrino, C., Zhang, C., Patras, P., Banchs, A., & Widmer, J. (2020). A machine-learning-based framework for optimizing the operation of future networks. IEEE Communications Magazine, 58(6), 20-25.

[19] Karthikeyan, P. (2020). Real-Time Data Warehousing: Performance Insights of Semi-Stream Joins Using Mongodb. International Journal of Management Research & Review, 10(4), 38-49

[20] Shahidinejad, A., & Ghobaei-Arani, M. (2020). Joint computation offloading and resource provisioning for e dge-cloud computing environment: A machine learning-based approach. Software: Practice and Experience, 50(12), 2212-2230.

[21] Mohan, R.S. (2020). Data-Driven Insights for Employee Retention: A Predictive Analytics Perspective. International Journal of Management Research & Review, 10(2), 44-59.

[22] Gill, S. S., Tuli, S., Toosi, A. N., Cuadrado, F., Garraghan, P., Bahsoon, R., ... & Buyya, R. (2020). ThermoSim: Deep learning-based framework for modeling and simulation of thermal-aware resource management for cloud computing environments. Journal of Systems and Software, 166, 110596.

[23] Sitaraman, S. R. (2020). Optimizing Healthcare Data Streams Using Real-Time Big Data Analytics and AI Techniques. International Journal of Engineering Research and Science & Technology, 16(3), 9-22.

[24] Gill, S. S., Tuli, S., Toosi, A. N., Cuadrado, F., Garraghan, P., Bahsoon, R., ... & Buyya, R. (2020). ThermoSim: Deep learning-based framework for modeling and simulation of thermal-aware resource management for cloud computing environments. Journal of Systems and Software, 166, 110596.

[25] Panga, N. K. R. (2020). Leveraging heuristic sampling and ensemble learning for enhanced insurance big data classification. International Journal of Financial Management (IJFM), 9(1).

[26] Mozaffari, S., Al-Jarrah, O. Y., Dianati, M., Jennings, P., & Mouzakitis, A. (2020). Deep learning-based vehicle behavior prediction for autonomous driving applications: A review. IEEE Transactions on Intelligent Transportation Systems, 23(1), 33-47.

[27] Gudivaka, R. L. (2020). Robotic Process Automation meets Cloud Computing: A Framework for Automated Scheduling in Social Robots. International Journal of Business and General Management (IJBGM), 8(4), 49-62.

[28] Niu, W., Zhang, X., Du, X., Zhao, L., Cao, R., & Guizani, M. (2020). A deep learning based static taint analysis approach for IoT software vulnerability location. Measurement, 152, 107139.

[29] Gudivaka, R. K. (2020). Robotic Process Automation Optimization in Cloud Computing Via Two-Tier MAC and LYAPUNOV Techniques. International Journal of Business and General Management (IJBGM), 9(5), 75-92.

[30] Miklosik, A., Kuchta, M., Evans, N., & Zak, S. (2019). Towards the adoption of machine learning-based analytical tools in digital marketing. Ieee Access, 7, 85705-85718.

[31] Devi, D. P. (2020). Artificial neural network enhanced real-time simulation of electric traction systems incorporating electro-thermal inverter models and FEA. International Journal of Engineering and Science Research, 10(3), 36-48.

[32] Zhu, H., Ge, W., & Liu, Z. (2019). Deep learning-based classification of weld surface defects. Applied Sciences, 9(16), 3312.

[33] Allur, N. S. (2020). Enhanced performance management in mobile networks: A big data framework incorporating DBSCAN speed anomaly detection and CCR efficiency assessment. Journal of Current Science, 8(4).

[34] Kumari, A., Vekaria, D., Gupta, R., & Tanwar, S. (2020, June). Redills: Deep learning-based secure data analytic framework for smart grid systems. In 2020 IEEE international conference on communications workshops (ICC Workshops) (pp. 1-6). IEEE.

[35] Deevi, D. P. (2020). Real-time malware detection via adaptive gradient support vector regression combined with LSTM and hidden Markov models. Journal of Science and Technology, 5(4).

[36] Souri, A., Mohammed, A. S., Potrus, M. Y., Malik, M. H., Safara, F., & Hosseinzadeh, M. (2020). Formal verification of a hybrid machine learning-based fault prediction model in Internet of Things applications. IEEE Access, 8, 23863-23874.

[37] Dondapati, K. (2020). Integrating neural networks and heuristic methods in test case prioritization: A machine learning perspective. International Journal of Engineering & Science Research, 10(3), 49–56.

[38] Han, T., Muhammad, K., Hussain, T., Lloret, J., & Baik, S. W. (2020). An efficient deep learning framework for intelligent energy management in IoT networks. IEEE Internet of Things Journal, 8(5), 3170-3179.

[39] Dondapati, K. (2020). Leveraging backpropagation neural networks and generative adversarial networks to enhance channel state information synthesis in millimeter-wave networks. International Journal of Modern Electronics and Communication Engineering, 8(3), 81-90

[40] Tuli, S., Basumatary, N., & Buyya, R. (2019, November). Edgelens: Deep learning-based object detection in integrated iot, fog and cloud computing environments. In 2019 4th International Conference on Information Systems and Computer Networks (ISCON) (pp. 496-502). IEEE.

[41] Gattupalli, K. (2020). Optimizing 3D printing materials for medical applications using AI, computational tools, and directed energy deposition. International Journal of Modern Electronics and Communication Engineering, 8(3).

[42] Shen, Z., Shang, X., Zhao, M., Dong, X., Xiong, G., & Wang, F. Y. (2019). A learning-based framework for error compensation in 3D printing. IEEE transactions on cybernetics, 49(11), 4042-4050.

[43] Allur, N. S. (2020). Big data-driven agricultural supply chain management: Trustworthy scheduling optimization with DSS and MILP techniques. Current Science & Humanities, 8(4), 1–16.

[44] Singh, G., Sharma, D., Goap, A., Sehgal, S., Shukla, A. K., & Kumar, S. (2019, October). Machine Learning based soil moisture prediction for Internet of Things based Smart Irrigation System. In 2019 5th International conference on signal processing, computing and control (ISPCC) (pp. 175-180). IEEE.

[45] Narla, S., Valivarthi, D. T., & Peddi, S. (2020). Cloud computing with artificial intelligence techniques: GWO-DBN hybrid algorithms for enhanced disease prediction in healthcare systems. Current Science & Humanities, 8(1), 14–30.

[46] Zhu, X., Luo, Y., Liu, A., Tang, W., & Bhuiyan, M. Z. A. (2020). A deep learning-based mobile crowdsensing scheme by predicting vehicle mobility. IEEE Transactions on Intelligent Transportation Systems, 22(7), 4648-4659.

[47] Kethu, S. S. (2020). AI and IoT-driven CRM with cloud computing: Intelligent frameworks and empirical models for banking industry applications. International Journal of Modern Electronics and Communication Engineering (IJMECE), 8(1), 54.

[48] Bedi, J., & Toshniwal, D. (2019). Deep learning framework to forecast electricity demand. Applied energy, 238, 1312-1326.

[49] Vasamsetty, C. (2020). Clinical decision support systems and advanced data mining techniques for cardiovascular care: Unveiling patterns and trends. International Journal of Modern Electronics and Communication Engineering, 8(2).

[50] Romeo, L., Loncarski, J., Paolanti, M., Bocchini, G., Mancini, A., & Frontoni, E. (2020). Machine learning-based design support system for the prediction of heterogeneous machine parameters in industry 4.0. Expert Systems with Applications, 140, 112869.

[51] Kadiyala, B. (2020). Multi-swarm adaptive differential evolution and Gaussian walk group search optimization for secured IoT data sharing using supersingular elliptic curve isogeny cryptography,International Journal of Modern Electronics and Communication Engineering,8(3).

[52] Sundaravadivel, P., Kesavan, K., Kesavan, L., Mohanty, S. P., & Kougianos, E. (2018). Smart-log: A deep-learning based automated nutrition monitoring system in the IoT. IEEE Transactions on Consumer Electronics, 64(3), 390-398.

[53] Valivarthi, D. T. (2020). Blockchain-powered AI-based secure HRM data management: Machine learning-driven predictive control and sparse matrix decomposition techniques. International Journal of Modern Electronics and Communication Engineering.8(4)

[54] Alli, A. A., & Alam, M. M. (2019). SecOFF-FCIoT: Machine learning based secure offloading in Fog-Cloud of things for smart city applications. Internet of Things, 7, 100070.

[55] Jadon, R. (2020). Improving AI-driven software solutions with memory-augmented neural networks, hierarchical multi-agent learning, and concept bottleneck models. International Journal of Information Technology and Computer Engineering, 8(2).

[56] Mao, H., Kathuria, D., Duffield, N., & Mohanty, B. P. (2019). Gap filling of high-resolution soil moisture for SMAP/Sentinel-1: A two-layer machine learning-based framework. Water Resources Research, 55(8), 6986-7009.

[57] Boyapati, S. (2020). Assessing digital finance as a cloud path for income equality: Evidence from urban and rural economies. International Journal of Modern Electronics and Communication Engineering (IJMECE), 8(3).

[58] Sharma, P., & Liu, H. (2020). A machine-learning-based data-centric misbehavior detection model for internet of vehicles. IEEE Internet of Things Journal, 8(6), 4991-4999.

[59] Gaius Yallamelli, A. R. (2020). A cloud-based financial data modeling system using GBDT, ALBERT, and Firefly algorithm optimization for high-dimensional generative topographic mapping. International Journal of Modern Electronics and Communication Engineering8(4).

[60] Li, L., Feng, H., Zhuang, W., Meng, N., & Ryder, B. (2017, September). Cclearner: A deep learning-based clone detection approach. In 2017 IEEE international conference on software maintenance and evolution (ICSME) (pp. 249-260). IEEE.

[61] Yalla, R. K. M. K., Yallamelli, A. R. G., & Mamidala, V. (2020). Comprehensive approach for mobile data security in cloud computing using RSA algorithm. Journal of Current Science & Humanities, 8(3).

[62] Da Costa, K. A., Papa, J. P., Lisboa, C. O., Munoz, R., & de Albuquerque, V. H. C. (2019). Internet of Things: A survey on machine learning-based intrusion detection approaches. Computer Networks, 151, 147-157.

[63] Samudrala, V. K. (2020). AI-powered anomaly detection for cross-cloud secure data sharing in multi-cloud healthcare networks. Journal of Current Science & Humanities, 8(2), 11–22.

[64] Kiran, R., Kumar, P., & Bhasker, B. (2020). DNNRec: A novel deep learning based hybrid recommender system. Expert Systems with Applications, 144, 113054.

[65] Ayyadurai, R. (2020). Smart surveillance methodology: Utilizing machine learning and AI with blockchain for bitcoin transactions. World Journal of Advanced Engineering Technology and Sciences, 1(1), 110–120.

[66] Wan, L., Sun, Y., Sun, L., Ning, Z., & Rodrigues, J. J. (2020). Deep learning based autonomous vehicle super resolution DOA estimation for safety driving. IEEE Transactions on Intelligent Transportation Systems, 22(7), 4301-4315.

[67] Chauhan, G. S., & Jadon, R. (2020). AI and ML-powered CAPTCHA and advanced graphical passwords: Integrating the DROP methodology, AES encryption, and neural network-based authentication for enhanced security. World Journal of Advanced Engineering Technology and Sciences, 1(1), 121–132.

[68] Masood, U., Farooq, H., & Imran, A. (2019, December). A machine learning based 3D propagation model for intelligent future cellular networks. In 2019 IEEE global communications conference (GLOBECOM) (pp. 1-6). IEEE.

[69] Narla, S. (2020). Transforming smart environments with multi-tier cloud sensing, big data, and 5G technology. International Journal of Computer Science Engineering Techniques, 5(1), 1-10.

[70] Liu, Y., Pang, Z., Karlsson, M., & Gong, S. (2020). Anomaly detection based on machine learning in IoT-based vertical plant wall for indoor climate control. Building and Environment, 183, 107212.

[71] Alavilli, S. K. (2020). Predicting heart failure with explainable deep learning using advanced temporal convolutional networks. International Journal of Computer Science Engineering Techniques, 5(2).

[72] Zhang, Y., Wang, H., Chen, W., Zeng, J., Zhang, L., & Wang, H. (2020). DP-GEN: A concurrent learning platform for the generation of reliable deep learning based potential energy models. Computer Physics Communications, 253, 107206.

[73] Qureshi, S. (2017). The forgotten awaken: ICT's evolving role in the roots of mass discontent. Information technology for development, 23(1), 1-17.

[74] Ran, M., Chen, L., & Li, W. (2020). Financial deepening, spatial spillover, and urban–rural income disparity: Evidence from China. Sustainability, 12(4), 1450.