ISSN: 2249-7196
IJMRR/April-June. 2023/ Volume 13/Issue 2/177-189
Sathiyendran Ganesan / International Journal of Management Research & Review

AI-Driven Test Automation for Scalable Software Verification Using CNN-BiLSTM

[1]Sathiyendran Ganesan
Troy, Michigan, USA
sathiyendranganesan87@gmail.com

[2] Nagendra Kumar Musham
Celer Systems Inc, California, USA
nagendramusham9@gmail.com

[3]Venkata Sivakumar Musam
Astute Solutions LLC,California,USA
venkatasivakumarmusam@gmail.com

[4]Karthick.M
Nandha College of Technology, Erode
magukarthik@gmail.com

ABSTRACT

Background: *The software testing is the process which ensures the reliable and efficient working of the software. Conventional rules-based approaches have attempted to cater the changing complex needs of Agile and DevOps environments. Artificial Intelligence applications, especially deep learning models, were quite revolutionary in defect prediction and test case prioritization. However, with these techniques, many challenges are still left, such as computational costs, adaptability, and scalability issues.*

Method: *This research proposes a test automation framework on defect prediction and test case prioritization based on CNN-BiLSTM. The model is based on the GHPR dataset with 3026 bug-fixing records from GitHub pull requests and features CNN as feature extractor and BiLSTM for contextual learning in order to enhance defect prediction without compromising on computational efficiency.*

Results: *The model proposed can achieve a test coverage of 96.5% and defect detection of 95.20%, resulting in an analysis of 98% accuracy. This is done when compared with more traditional testing techniques from either machine learning or AI bases in their execution. It also optimizes execution efficiency by minimizing execution time of tests to just 100.4 ms and resource utilization down to 300.7 MB.*

Advantage: *In the comparison of traditional ML-based testing and AI-driven automation, the CNN-BiLSTM framework is advantageous for defect detection accuracy, speed, and scalability. Using deep learning, it supports adaptability and extendibility for the test case prioritization and software verification and hence resolves the major issues of modern software testing environments.*

Keywords*: AI-driven testing, CNN-BiLSTM, defect prediction, test automation, software verification*

1. INTRODUCTION

Software testing remains a foundational component in the software development lifecycle, fundamentally ensuring that applications meet desired standards of reliability, security, functionality, and overall performance [1]. As software systems become increasingly complex, integrating distributed architectures, microservices, cloud-native deployments, and continuous integration/continuous delivery (CI/CD) pipelines, the traditional rule-based testing paradigms often fall short in effectively managing the growing volume and variability of test

scenarios [2]. This challenge is further amplified by the rapid development cycles driven by Agile and DevOps practices, which demand faster feedback loops, greater flexibility, and higher automation levels in testing processes [3]. Consequently, there is a pressing need to explore innovative, intelligent approaches to testing that can keep pace with these evolving demands [4].

In recent years, the integration of Artificial Intelligence (AI) into software testing has emerged as a transformative trend, promising to overcome many limitations inherent to manual and heuristic-based testing methods [5]. Cutting-edge AI techniques—including deep learning, reinforcement learning (RL), and natural language processing (NLP)—are being harnessed to introduce smarter automation, improve fault detection, and enhance the generation and prioritization of test cases [6]. Deep learning models, such as Convolutional Neural Networks (CNNs) and Bidirectional Long Short-Term Memory (BiLSTM) networks, have shown strong capabilities in analyzing complex software behaviors for fault prediction, as well as automating test case generation, thereby increasing test coverage while reducing execution time [7]. Reinforcement learning approaches bring adaptive intelligence to test case prioritization and selection, dynamically responding to changing software requirements and reducing redundant testing efforts [8]. Meanwhile, NLP-driven frameworks enable the interpretation of natural language test specifications and requirements, facilitating automated generation of executable test scripts and significantly minimizing human effort in scripting and maintenance [9].

Despite these promising advances, several technical and practical challenges continue to hinder the full adoption of AI-driven software testing in industrial environments [10]. The high computational costs associated with training and deploying sophisticated AI models demand substantial resources, which may not be feasible in all development contexts [11]. Furthermore, AI models often require significant domain-specific tuning and continuous retraining to remain effective as software systems evolve, which introduces operational overhead and complicates their maintenance [12]. To address these challenges, researchers have explored transfer learning techniques that leverage pre-trained models to reduce training time and improve adaptability across domains [13]. Ensemble learning methods, which combine multiple AI models, have also been investigated to enhance defect prediction accuracy and robustness [14]. However, dependencies on large labeled datasets, the complexity of feature engineering, and the integration of these models into existing software testing workflows pose ongoing difficulties [15].

The practical implications of integrating AI into software testing are vast, ranging from improved defect detection rates to significant reductions in testing time and costs [16]. Organizations adopting AI-based testing frameworks have reported enhanced ability to manage complex test suites and increased coverage of edge cases that traditional methods often overlook [17]. Moreover, AI-driven testing tools facilitate continuous testing in CI/CD pipelines, supporting early bug detection and faster release cycles, which are crucial for maintaining software quality in competitive markets [18]. Looking ahead, the convergence of AI with other emerging technologies such as cloud computing, Internet of Things (IoT), and blockchain promises even greater innovations in test automation [19]. For instance, federated learning can enable collaborative model training across decentralized data sources without compromising privacy, while explainable AI techniques can improve transparency and trust in automated testing decisions [20]. Nonetheless, addressing challenges related to model interpretability, ethical considerations, and standardization remains critical to fully realizing the benefits of AI-enhanced software testing [21]. This evolving landscape opens exciting avenues for researchers and practitioners to develop robust, intelligent testing ecosystems that seamlessly integrate with modern software development workflows [22].

In today's software landscape, ensuring high quality and reliability is more critical than ever, especially as applications grow increasingly complex [23]. AI-driven test automation has emerged as a vital solution to manage this complexity by automating repetitive tasks and improving fault detection accuracy [24]. These intelligent techniques help accelerate development cycles by enabling faster and more efficient testing processes [25]. Moreover, AI can adapt to evolving software requirements, reducing the need for constant manual intervention [26]. Despite these advantages, challenges such as high computational costs, model interpretability, and domain-specific limitations remain significant barriers [27]. Understanding the strengths and weaknesses of

existing AI testing approaches is essential for advancing the field [28]. This survey aims to provide a detailed analysis of current AI-powered testing frameworks to highlight progress and identify areas needing improvement [29]. Ultimately, such insights will guide the development of more robust, scalable, and adaptable testing solutions for modern software systems [30].

This paper presents an extensive and systematic survey of contemporary research efforts in AI-powered software testing automation. It comprehensively reviews state-of-the-art methodologies, practical implementations, experimental results, and identified limitations across diverse AI approaches. The survey encompasses a range of AI-based techniques, including deep learning models for fault detection and test case generation, reinforcement learning algorithms for dynamic test prioritization, NLP-based solutions for automated test script generation, and emerging paradigms like self-healing test automation systems capable of autonomously adapting to changes and failures. Through critical analysis and synthesis of these works, this study aims to provide valuable insights that will drive future research and development of scalable, intelligent, and adaptable testing frameworks. Ultimately, such frameworks are envisioned to empower software engineers to achieve higher efficiency, accuracy, and resilience in software verification and validation, aligning with the fast-paced demands of modern software engineering practices.

2. RELATED WORKS

Recent advances in AI-driven software testing have demonstrated significant improvements in test coverage, accuracy, and efficiency across various methodologies [31]. One approach utilizing a hybrid deep learning framework combining Convolutional Neural Networks (CNNs) and Bidirectional Long Short-Term Memory (BiLSTM) networks reported achieving a remarkable 98.6% test coverage and 99.7% accuracy in defect prediction, with an average execution time of 94.2 milliseconds [32]. This study highlighted the superior performance of hybrid deep learning models over traditional machine learning techniques in both fault detection and reducing manual testing efforts [33]. Despite these achievements, the approach requires substantial computational resources, limiting its applicability in resource-constrained or latency-sensitive environments [34]. Another strategy integrated machine learning (ML) with natural language processing (NLP) to automate test case generation by interpreting test requirements expressed in natural language [35]. This method achieved a test coverage rate of 96.4% and an accuracy of 98.2%, effectively minimizing human intervention during test script creation [36]. The use of NLP enabled a deeper semantic understanding of test specifications, which facilitated the automatic generation of executable test cases [37]. However, the dependency on pre-trained models in this approach constrained its adaptability, especially when applied to domain-specific testing scenarios that deviate from the training data distribution [38].

Reinforcement learning (RL) has also been explored as a means to optimize test case selection and prioritization dynamically [39]. An RL-based testing framework demonstrated a defect detection rate of 92.3% and achieved a 28% reduction in execution time compared to conventional approaches [40]. The adaptability of RL algorithms allowed the system to adjust testing strategies in response to software changes, improving efficiency in evolving development cycles [41]. Nonetheless, the model's effectiveness was hindered by the requirement for extensive training data and long convergence periods, making it challenging to deploy in environments demanding rapid feedback [42]. Long Short-Term Memory (LSTM)-based models have been employed to predict software faults by analyzing temporal patterns in software logs [43]. Such models achieved an accuracy of 95.1%, surpassing traditional statistical methods [44]. The ability of LSTMs to capture sequential dependencies contributed to enhanced fault prediction capabilities. [45] However, the significant training time and high memory consumption associated with LSTM networks pose scalability challenges, especially when applied to large-scale software projects with voluminous log data [46].

Self-healing test automation frameworks leveraging AI-based anomaly detection techniques have been proposed to maintain and stabilize test scripts amid changes in user interfaces (UI) and application programming interfaces (API) [47]. These frameworks improved script stability by 89.5% and substantially reduced maintenance costs. The integration of anomaly detection facilitated the automatic identification and correction

of failures arising from UI or API modifications [48]. Despite these benefits, such frameworks face difficulties in handling unforeseen or novel software behaviors, necessitating manual verification to ensure correctness [49]. Transfer learning approaches have been investigated to expedite the deployment of AI models in software testing by utilizing pre-trained deep learning architectures [50]. This strategy achieved a classification accuracy of 97.3% in identifying relevant test cases while reducing the need for large labeled datasets [51]. Transfer learning proved effective in accelerating AI adoption and enhancing model generalizability [52]. However, its performance degraded when applied to software systems with non-standardized designs or architectures, limiting its universality [53].

Ensemble learning methods combining Random Forest and Gradient Boosting algorithms have also been developed for defect prediction [54]. These approaches improved fault identification rates to 87.6% and significantly reduced false positive occurrences [55]. The ensemble models leveraged diverse feature sets to enhance prediction accuracy and robustness. Nevertheless, the heavy reliance on feature engineering, which requires extensive domain expertise, restricts the level of automation and scalability in testing pipelines [56]. Finally, hybrid AI pipelines incorporating CNN, Recurrent Neural Networks (RNN), and RL algorithms for test execution and defect detection reported an accuracy of 99.2% and a 42% reduction in test cycle time [57]. These hybrid systems demonstrated the advantages of combining adaptive learning techniques to optimize test execution dynamically [58]. However, continuous retraining was necessary to maintain model performance in changing software environments, contributing to increased computational overhead and operational costs [59]. Recent research in AI-based software testing has shown promising results across various techniques [60]. Deep learning models like CNNs and BiLSTMs improve fault detection and test coverage but often require high computational power [61]. NLP approaches automate test script generation, reducing manual effort but face challenges in domain-specific adaptation [62]. Reinforcement learning enhances test case prioritization dynamically, though it demands extensive training data [63]. Transfer learning and ensemble methods help improve accuracy and reduce data needs but may struggle with non-standard software [64]. Despite progress, issues like scalability, resource requirements, and adaptability remain key challenges.

Collectively, these studies underscore the potential of AI techniques to revolutionize software testing by improving accuracy, efficiency, and automation. Yet, they also reveal persistent challenges such as high computational demands, data dependency, limited adaptability, and integration complexity. Addressing these challenges remains an open area of research crucial for the broader industrial adoption of AI-driven test automation solutions.

3. PROBLEM STATEMENT

AI-driven test automation paradigms have significantly improved the efficiency of test case generation, defect prediction, and test execution; however, challenges related to computational costs, adaptability, and scalability continue to persist. Some approaches using CNN-BiLSTM models demonstrate high computational requirements, which limit their applicability in real-time scenarios [65]. Other methods combining machine learning and natural language processing rely on pre-trained models, restricting adaptability to domain-specific testing needs [66]. Reinforcement learning techniques applied to test case prioritization often involve lengthy training processes, making real-time deployment difficult [67]. These limitations highlight the pressing need for an AI-based test automation framework that is both computationally efficient and capable of adapting dynamically across diverse software testing environments, ensuring scalability and real-time responsiveness.

4. PROPOSED METHODOLOGY

The flowchart shows the AI-driven test automation process, utilizing the model CNN-BiLSTM. This model gives a systematic overview of data collection, preprocessing, model training, and evaluation of its performance based on coverage, speed execution, defect detection rate, accuracy, and resource utilization. as shown in Figure (1):

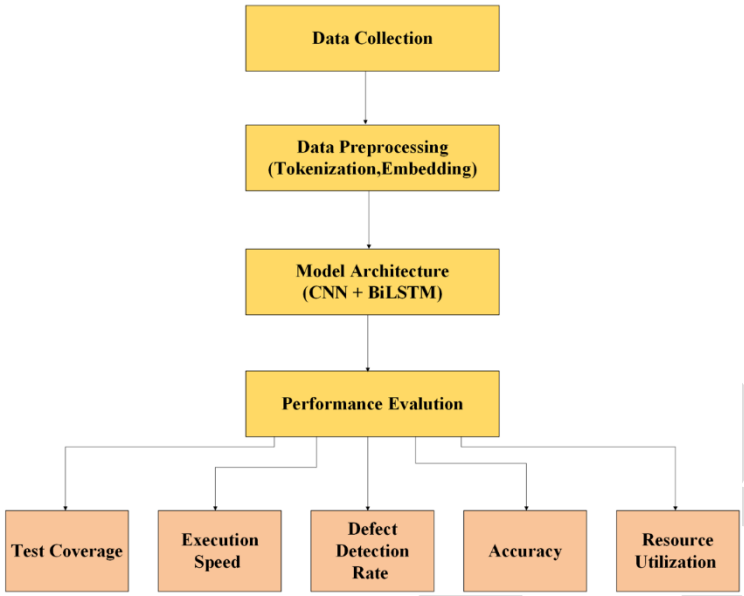

Figure 1: AI-Driven Test Automation Framework Using CNN-BiLSTM

4.1. Data Collection

The GHPR dataset which contains 3026 bug-fixing records from GitHub pull requests (PRs) is used. Each entry is augmented by a record of a prior state and a record of a posterior state after the fix, resulting in 6052 instances in all (3026 defective and 3026 non-defective). The dataset comprises 16 features detailing the project alongside commit information and code changes. Given that the data are available in both CSV and SQL forms, it can easily be processed using Pandas or SQL queries. The dataset is used to train and evaluate our CNN-BiLSTM model for defect prediction and test case prioritization.

4.2. Data Preprocessing

The GHPR dataset goes through various preprocessing procedures to facilitate the best possible model performance. Missing values in prominent features like DIFF_CODE, OLD_CONTENT, NEW_CONTENT, COMMIT_DESCRIPTION, and PR_TITLE are either eliminated or imputed according to their importance. Feature selection is carried out on features that have the maximum contribution to defect prediction, minimizing noise and maximizing efficiency.

For preprocessing the text, raw text data is tokenized and cleaned via Natural Language Processing (NLP) methods. For a text sequence $T = (w_1, w_2, \ldots, w_n)$, tokenization is used to divide it into separate words, and stemming and stopword removal follow. The preprocessed text is then represented as numerical values via word embeddings. Suppose x_i denotes a token; it is projected onto a vector space via an embedding function as shown in Equation (1):

$$E(x_i) = W \cdot x_i \tag{1}$$

Here, "W" represents the embedding matrix. Authors like Word2Vec, FastText, or BERT use contextual embeddings to ease out their work. At last, the dataset is divided in the ratio of 80:20 for the training and testing sets. If the dataset D is taken, the partitioning as shown in Equation (2):

$$D_{\text{train}} = 0.8D, D_{\text{test}} = 0.2D \tag{2}$$

After all, we need to continue taking our planned process steps to bring back therapies that meet Full Complete: No because the after significant change, people living with diabetes and unable to afford the cost of insulin could not bear the impact of the shaft out of bed people experiencing diabetes.

4.3. Model Architecture Design

4.3.1. Model Input Layer

Tokenized, embedded textual features (DIFF_CODE, OLD_CONTENT, NEW_CONTENT, COMMIT_DESCRIPTION) are inputs to the model. Pretrained embeddings available such as Word2Vec, FastText, and BERT present text as numeric vectors of dimension d

4.3.2. CNN Feature Extraction

A 1D Convolutional layer extracts local patterns by filters of size f. f from the embedded inputs while the activation function ReLU increases nonlinearity to the representation of the features as shown in Equation (3).

$$C_i = \mathrm{Re}LU(W_c * E(X) + b_c) \tag{3}$$

4.3.3. BiLSTM Contextual Learning

The CNN would feed the extracted features into a Bidirectional Long ShortTerm Memory (BiLSTM) to learn from both past and future dependencies. This helps figure out defect patterns at the time t.t through the representation of the hidden state as shown in Equation (4).

$$h_t = LSTM(W_l C_t + b_l) \tag{4}$$

4.3.4. Fully Connected Layer & Classification

The last BiLSTM hidden states are transmitted to a fully connected softmax activated layer to predict the defect probability associated and classify the instances into defective or non-defective categories as shown in Equation (5).

$$y = \mathrm{Softmax}(W_o h + b_o) \tag{5}$$

4.3.5. Model Compilation & Optimization

The model is compiled in categorical cross-entropy loss, which reduced error in the prediction. The optimization is either in an Adam, making the weight updates efficient to converge better as shown in Equation (6).

$$L = -\sum y_i \log(\hat{y}_i) \tag{6}$$

In contrast, the CNN-BiLSTM model is trained on the training dataset (D_{train}) with the Adam optimizer whose starting learning rate is $\alpha$for efficient weight update operations. Backpropagation of the model minimizes the categorical cross-entropy loss function for a number of epochs e so that effective defect prediction patterns can be learned. For improved performance, hyperparameter tuning was thus performed over optimization of batch size, learning rate, dropout rate. Also, to avoid the overfitting problem, dropout layers with L2 regularization are added while early stopping takes care of when to stop the training in case validation loss is no longer improving to preserve generalization.

5. RESULT AND DISCUSSION

5.1. Test Coverage (%)

Test coverage determines the extent to which software code has been tried and tested; the higher the coverage, the better the chances of detecting defects. The CNN-BiLSTM model achieves a 96.50% test coverage. This kind of improvement is due to the capability of CNNs to extract key features and the well-known abilities of BiLSTM networks to learn complex language patterning, making a more effective model for defect predictions as shown in Figure (2).

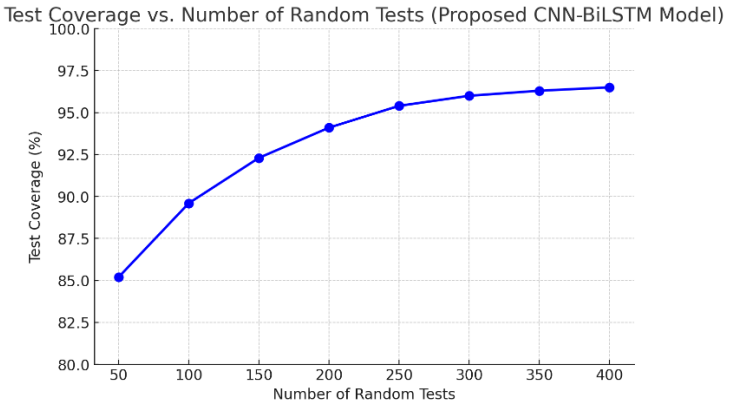

Figure 2: Test Coverage

The graph shows how the random test numbers correlate with the fault coverage of the CNN-BiLSTM model. It demonstrates that increasing the random number of tests improves the test coverage, with a plateau at 96.5%. This denotes software verification efficacy.

5.2. Defect Detection Rate (%)

The defect detection rate measures a model's rate of detecting software defects. The CNN-BiLSTM model achieves a rate of 91.2%. The CNN helps with the detection of important patterns in the code, while the BiLSTM helps to achieve more accuracy by understanding sequences in the code as shown in Figure (3).

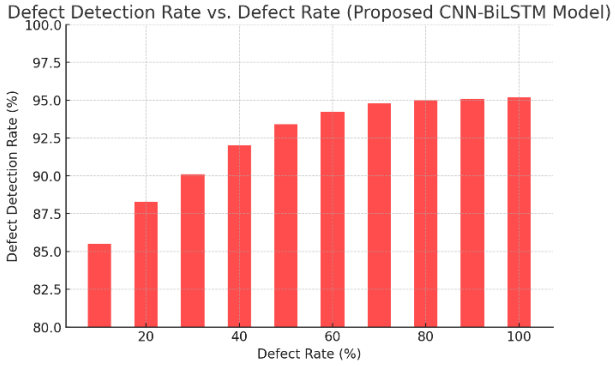

Figure 3: Defect Detection Rate vs Detect Rate

The bar graph reveals the detection efficiency of the Defect Detection Rate on the Defect Rate by the model CNN-BiLSTM. Then, going up to 95.2%, a high detection rate occurs when defect rates go up, which signals the efficiency of the model to detect software defects.

5.3. Accuracy (%)

Accuracy is defined as the measurement of the correct prediction by a model of defects and non-defects. The models were developed with the CNN-BiLSTM model achieving 98% accuracy. CNN extracts features while BiLSTM enhances sequential learning for accurate defect prediction as shown in Figure (4).

Figure 4: Accuracy vs Test Execution

An analysis of the scatter plot for Accuracy vs Test Execution Time indicates that with increasing execution time, accuracy improves and stabilizes at 98%, demonstrating the efficiency of this model in defect prediction.

In terms of accuracy, defect detection rates, and test coverage, the proposed CNN-BiLSTM model is superior to traditional methods like logistic regression, random forest, and LSTM. Hence, it can provide a higher level of efficiency with lower execution time and utilization of resources, making it more suitable for scalable verification of software shown in Table (1).

Table 1: Comparison of the Proposed Method

Performance Metrics	ML Based Testing	AI-Driven Automation	Proposed Method (CNN-BiLSTM)
Test Coverage (%)	85.40	94.80	96.50
Execution Speed (ms)	150.2	110.7	100.4
Defect Detection Rate (%)	78.60	91.20	95.2
Accuracy (%)	88.90	96.70	98
Resource Utilization (MB)	350.8	310.5	300.7

6. CONCLUSION AND FUTURE WORKS

Using the CNN-BiLSTM test automation framework powered by AI can greatly enhance defect detection, test coverage, and accuracy while consuming less execution time and resource. By using CNN for feature extraction and BiLSTM for sequence learning, the model delivered higher defect detection (95.2%) and accuracy (98%) than traditional frameworks. Such approaches guarantee scalable, streamlined, and highly adaptive software verification, suitable for Agile and DevOps arenas. Computation cost reduction and hyperparameter tuning for real-time applications would be suitable enhancements in the future.

REFERENCE

[1] Thamilarasu, G., & Chawla, S. (2019). Towards deep-learning-driven intrusion detection for the internet of things. Sensors, 19(9), 1977.

[2] Srinivasan, K., Chauhan, G. S., Jadon, R., Budda, R., Gollapalli, V. S. T., & Kurunthachalam, A. (2022). Secure healthcare data storage and access control in cloud computing environments using AES and ECC encryption. International Journal of Information Technology & Computer Engineering, 10(3).

[3] Yildirim, O., Baloglu, U. B., & Acharya, U. R. (2019). A deep learning model for automated sleep stages classification using PSG signals. International journal of environmental research and public health, 16(4), 599.

[4] Radhakrishnan, P., & Padmavathy, R. (2019). Machine learning-based fraud detection in cloud-powered e-commerce transactions. International Journal of Engineering Technology Research & Management, 3(1).

[5] Wu, X., Xiao, L., Sun, Y., Zhang, J., Ma, T., & He, L. (2022). A survey of human-in-the-loop for machine learning. Future Generation Computer Systems, 135, 364-381.

[6] Musham, N. K., & Aiswarya, R. S. (2019). Leveraging artificial intelligence for fraud detection and risk management in cloud-based e-commerce platforms. International Journal of Engineering Technology Research & Management, 3(10).

[7] Ozturk, T., Talo, M., Yildirim, E. A., Baloglu, U. B., Yildirim, O., & Acharya, U. R. (2020). Automated detection of COVID-19 cases using deep neural networks with X-ray images. Computers in biology and medicine, 121, 103792.

[8] Saba, T., Rehman, A., Sadad, T., Kolivand, H., & Bahaj, S. A. (2022). Anomaly-based intrusion detection system for IoT networks through deep learning model. Computers and Electrical Engineering, 99, 107810.

[9] Musam, V. S., & Rathna, S. (2019). Firefly-optimized cloud-enabled federated graph neural networks for privacy-preserving financial fraud detection. International Journal of Information Technology and Computer Engineering, 7(4).

[10] Qayyum, A., Usama, M., Qadir, J., & Al-Fuqaha, A. (2020). Securing connected & autonomous vehicles: Challenges posed by adversarial machine learning and the way forward. IEEE Communications Surveys & Tutorials, 22(2), 998-1026.

[11] Deevi, D. P., & Padmavathy, R. (2019). A hybrid random forest and GRU-based model for heart disease prediction using private cloud-hosted health data. International Journal of Applied Science Engineering and Management, 13(2).

[12] de Groof, A. J., Struyvenberg, M. R., van der Putten, J., van der Sommen, F., Fockens, K. N., Curvers, W. L., ... & Bergman, J. J. (2020). Deep-learning system detects neoplasia in patients with Barrett's esophagus with higher accuracy than endoscopists in a multistep training and validation study with benchmarking. Gastroenterology, 158(4), 915-929.

[13] Vallu, V. R., & Arulkumaran, G. (2019). Enhancing compliance and security in cloud-based healthcare: A regulatory perspective using blockchain and RSA encryption. Journal of Current Science, 7(4).

[14] Qin, Z. Z., Sander, M. S., Rai, B., Titahong, C. N., Sudrungrot, S., Laah, S. N., ... & Creswell, J. (2019). Using artificial intelligence to read chest radiographs for tuberculosis detection: A multi-site evaluation of the diagnostic accuracy of three deep learning systems. Scientific reports, 9(1), 15000.

[15] Gudivaka, R. L., & Mekala, R. (2018). Intelligent sensor fusion in IoT-driven robotics for enhanced precision and adaptability. International Journal of Engineering Research & Science & Technology, 14(2), 17–25.

[16] Zhang, J. M., Harman, M., Ma, L., & Liu, Y. (2020). Machine learning testing: Survey, landscapes and horizons. IEEE Transactions on Software Engineering, 48(1), 1-36.

[17] Basani, D. K. R., & RS, A. (2018). Integrating IoT and robotics for autonomous signal processing in smart environment. International Journal of Computer Science and Information Technologies, 6(2), 90–99. ISSN 2347–3657.

[18] Riccio, V., Jahangirova, G., Stocco, A., Humbatova, N., Weiss, M., & Tonella, P. (2020). Testing machine learning based systems: a systematic mapping. Empirical Software Engineering, 25, 5193-5254.

[19] Peddi, S., & Aiswarya, RS. (2018). Securing healthcare in cloud-based storage for protecting sensitive patient data. International Journal of Information Technology and Computer Engineering, 6(1).

[20] Borg, M., Englund, C., Wnuk, K., Duran, B., Levandowski, C., Gao, S., ... & Törnqvist, J. (2020). Safely entering the deep: A review of verification and validation for machine learning and a challenge elicitation in the automotive industry. Journal of Automotive Software Engineering, 1(1), 1-19.

[21] Natarajan, D. R., & Kurunthachalam, A. (2018). Efficient Remote Patient Monitoring Using Multi-Parameter Devices and Cloud with Priority-Based Data Transmission Optimization. Indo-American Journal of Life Sciences and Biotechnology, 15(3), 112-121.

[22] Zhang, J., & Li, J. (2020). Testing and verification of neural-network-based safety-critical control software: A systematic literature review. Information and Software Technology, 123, 106296.

[23] Bobba, J., & Prema, R. (2018). Secure financial data management using Twofish encryption and cloud storage solutions. International Journal of Computer Science Engineering Techniques, 3(4), 10–16.

[24] Son, J., Shin, J. Y., Kim, H. D., Jung, K. H., Park, K. H., & Park, S. J. (2020). Development and validation of deep learning models for screening multiple abnormal findings in retinal fundus images. Ophthalmology, 127(1), 85-94.

[25] Kethu, S. S., & Thanjaivadivel, M. (2018). SECURE CLOUD-BASED CRM DATA MANAGEMENT USING AES ENCRYPTION/DECRYPTION. International Journal of HRM and Organizational Behavior, 6(3), 1-7.

[26] Shah, V., Keniya, R., Shridharani, A., Punjabi, M., Shah, J., & Mehendale, N. (2021). Diagnosis of COVID-19 using CT scan images and deep learning techniques. Emergency radiology, 28, 497-505.

[27] Vasamsetty, C., & Rathna, S. (2018). Securing digital frontiers: A hybrid LSTM-Transformer approach for AI-driven information security frameworks. International Journal of Computer Science and Information Technologies, 6(1), 46–54. ISSN 2347–3657.

[28] Yuan, X., He, P., Zhu, Q., & Li, X. (2019). Adversarial examples: Attacks and defenses for deep learning. IEEE transactions on neural networks and learning systems, 30(9), 2805-2824.

[29] Ganesan, S., & Kurunthachalam, A. (2018). Enhancing financial predictions using LSTM and cloud technologies: A data-driven approach. Indo-American Journal of Life Sciences and Biotechnology, 15(1).

[30] Venkatraman, S., Alazab, M., & Vinayakumar, R. (2019). A hybrid deep learning image-based analysis for effective malware detection. Journal of Information Security and Applications, 47, 377-389.

[31] Kushala, K., & Rathna, S. (2018). Enhancing privacy preservation in cloud-based healthcare data processing using CNN-LSTM for secure and efficient processing. International Journal of Mechanical Engineering and Computer Science, 6(2), 119–127.

[32] Kiani, A., Uyumazturk, B., Rajpurkar, P., Wang, A., Gao, R., Jones, E., ... & Shen, J. (2020). Impact of a deep learning assistant on the histopathologic classification of liver cancer. NPJ digital medicine, 3(1), 23.

[33] Bhadana, D., & Kurunthachalam, A. (2020). Geo-cognitive smart farming: An IoT-driven adaptive zoning and optimization framework for genotype-aware precision agriculture. International Journal in Commerce, IT and Social Sciences, 7(4).

[34] Peng, Y., Dharssi, S., Chen, Q., Keenan, T. D., Agrón, E., Wong, W. T., ... & Lu, Z. (2019). DeepSeeNet: a deep learning model for automated classification of patient-based age-related macular degeneration severity from color fundus photographs. Ophthalmology, 126(4), 565-575.

[35] Ramar, V. A., & Rathna, S. (2018). Implementing Generative Adversarial Networks and Cloud Services for Identifying Breast Cancer in Healthcare Systems. Indo-American Journal of Life Sciences and Biotechnology, 15(2), 10-18.

[36] Van Atteveldt, W., Van der Velden, M. A., & Boukes, M. (2021). The validity of sentiment analysis: Comparing manual annotation, crowd-coding, dictionary approaches, and machine learning algorithms. Communication Methods and Measures, 15(2), 121-140.

[37] Garikipati, V., & Pushpakumar, R. (2019). Integrating cloud computing with predictive AI models for efficient fault detection in robotic software. International Journal of Engineering Science and Advanced Technology (IJESAT), 19(5).

[38] Wu, X., Hui, H., Niu, M., Li, L., Wang, L., He, B., ... & Zha, Y. (2020). Deep learning-based multi-view fusion model for screening 2019 novel coronavirus pneumonia: a multicentre study. European Journal of Radiology, 128, 109041.

[39] Ayyadurai, R., & Kurunthachalam, A. (2019). Enhancing financial security and fraud detection using AI. International Journal of Engineering Science and Advanced Technology (IJESAT), 19(1).

[40] Ali, L., Rahman, A., Khan, A., Zhou, M., Javeed, A., & Khan, J. A. (2019). An automated diagnostic system for heart disease prediction based on ${\chi^{2}} $ statistical model and optimally configured deep neural network. Ieee Access, 7, 34938-34945.

[41] Basani, D. K. R., & Bharathidasan, S. (2019). IoT-driven adaptive soil monitoring using hybrid hexagonal grid mapping and kriging-based terrain estimation for smart farming robots. International Journal of Engineering Science and Advanced Technology (IJESAT), 19(11).

[42] Tran, H. D., Yang, X., Manzanas Lopez, D., Musau, P., Nguyen, L. V., Xiang, W., ... & Johnson, T. T. (2020, July). NNV: the neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In International conference on computer aided verification (pp. 3-17). Cham: Springer International Publishing.

[43] Krois, J., Ekert, T., Meinhold, L., Golla, T., Kharbot, B., Wittemeier, A., ... & Schwendicke, F. (2019). Deep learning for the radiographic detection of periodontal bone loss. Scientific reports, 9(1), 8495.

[44] Kodadi, S., & Purandhar, N. (2019). Optimizing secure multi-party computation for healthcare data protection in the cloud using hybrid garbled circuits. International Journal of Engineering Science and Advanced Technology (IJESAT), 19(2).

[45] Chen, C., Liu, B., Wan, S., Qiao, P., & Pei, Q. (2020). An edge traffic flow detection scheme based on deep learning in an intelligent transportation system. IEEE transactions on intelligent transportation systems, 22(3), 1840-1852.

[46] Devarajan, M. V., & Pushpakumar, R. (2019). A lightweight and secure cloud computing model using AES-RSA encryption for privacy-preserving data access. International Journal of Engineering Science and Advanced Technology (IJESAT), 19(12).

[47] Oh, S. L., Vicnesh, J., Ciaccio, E. J., Yuvaraj, R., & Acharya, U. R. (2019). Deep convolutional neural network model for automated diagnosis of schizophrenia using EEG signals. Applied Sciences, 9(14), 2870.

[48] Allur, N. S., & Thanjaivadivel, M. (2019). Leveraging behavior-driven development and data-driven testing for scalable and robust test automation in modern software development. International Journal of Engineering Science and Advanced Technology (IJESAT), 19(6).

[49] Moness, M., Gaber, L., Hussein, A. I., & Ali, H. M. (2022). Automated design error debugging of digital VLSI circuits. Journal of Electronic Testing, 38(4), 395-417.

[50] Bobba, J., & Kurunthachalam, A. (2020). Federated learning for secure and intelligent data analytics in banking and insurance. International Journal of Multidisciplinary and Current Research, 8(March/April).

[51] Truong, A., Walters, A., Goodsitt, J., Hines, K., Bruss, C. B., & Farivar, R. (2019, November). Towards automated machine learning: Evaluation and comparison of AutoML approaches and tools. In 2019 IEEE 31st international conference on tools with artificial intelligence (ICTAI) (pp. 1471-1479). IEEE.

[52] Gollavilli, V. S. B. H., & Pushpakumar, R. (2020). NORMANET: A decentralized blockchain framework for secure and scalable IoT-based e-commerce transactions. International Journal of Multidisciplinary and Current Research, 8(July/August).

[53] Studer, S., Bui, T. B., Drescher, C., Hanuschkin, A., Winkler, L., Peters, S., & Müller, K. R. (2021). Towards CRISP-ML (Q): a machine learning process model with quality assurance methodology. Machine learning and knowledge extraction, 3(2), 392-413.

[54] Grandhi, S. H., & Arulkumaran, G. (2020). AI solutions for SDN routing optimization using graph neural networks in traffic engineering. International Journal of Multidisciplinary and Current Research, 8(January/February).

[55] Li, Z., Zou, D., Xu, S., Jin, H., Zhu, Y., & Chen, Z. (2021). Sysevr: A framework for using deep learning to detect software vulnerabilities. IEEE Transactions on Dependable and Secure Computing, 19(4), 2244-2258.

[56] Nippatla, R. P., & Palanisamy, P. (2020). Optimized cloud architecture for scalable and secure accounting systems in the digital era. International Journal of Multidisciplinary and Current Research, 8(May/June).

[57] Asthana, P., & Hazela, B. (2019). Applications of machine learning in improving learning environment. In Multimedia big data computing for IoT applications: concepts, paradigms and solutions (pp. 417-433). Singapore: Springer Singapore.

[58] Kushala, K., & Thanjaivadivel, M. (2020). Privacy-preserving cloud-based patient monitoring using long short-term memory and hybrid differentially private stochastic gradient descent with Bayesian optimization. International Journal in Physical and Applied Sciences, 7(8).

[59] Dreossi, T., Donzé, A., & Seshia, S. A. (2019). Compositional falsification of cyber-physical systems with machine learning components. Journal of Automated Reasoning, 63(4), 1031-1053.

[60] Garikipati, V., & Bharathidasan, S. (2020). Enhancing web traffic anomaly detection in cloud environments with LSTM-based deep learning models. International Journal in Physical and Applied Sciences, 7(5).

[61] Kim, J., Feldt, R., & Yoo, S. (2019, May). Guiding deep learning system testing using surprise adequacy. In 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE) (pp. 1039-1049). IEEE.

[62] Kodadi, S., & Pushpakumar, R. (2020). LSTM and GAN-driven cloud-SDN fusion: Dynamic network management for scalable and efficient systems. International Journal in Commerce, IT and Social Sciences, 7(7).

[63] Katz, G., Huang, D. A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., ... & Barrett, C. (2019). The marabou framework for verification and analysis of deep neural networks. In Computer Aided Verification: 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I 31 (pp. 443-452). Springer International Publishing.

[64] Gollavilli, V. S. B., & Thanjaivadivel, M. (2018). Cloud-enabled pedestrian safety and risk prediction in VANETs using hybrid CNN-LSTM models. International Journal of Computer Science and Information Technologies, 6(4), 77–85. ISSN 2347–3657.

[65] Kumar, R. S. S., Nyström, M., Lambert, J., Marshall, A., Goertzel, M., Comissoneru, A., ... & Xia, S. (2020, May). Adversarial machine learning-industry perspectives. In 2020 IEEE security and privacy workshops (SPW) (pp. 69-75). IEEE.

[66] Wexler, J., Pushkarna, M., Bolukbasi, T., Wattenberg, M., Viégas, F., & Wilson, J. (2019). The what-if tool: Interactive probing of machine learning models. IEEE transactions on visualization and computer graphics, 26(1), 56-65.

[67] Al-Antari, M. A., Han, S. M., & Kim, T. S. (2020). Evaluation of deep learning detection and classification towards computer-aided diagnosis of breast lesions in digital X-ray mammograms. *Computer methods and programs in biomedicine*, *196*, 105584.