International Journal of Multidisciplinary and Current Research ISSN: 2321-3124
Research Article Available at: http://ijmcr.com

AI-Driven Software Testing and Development: Enhancing Automation, Efficiency, and Reliability in Agile and DevOps Environments

[1*]Sathiyendran Ganesan and [2]G. Arulkumaran

[1]Troy, Michigan, USA
[2]Bule Hora University, Bule Hora, Oromia, ET,

Received 01 April 2021, Accepted 20 April 2021, Available online 25 April 2021, **Vol.9 (March/April 2021 issue)**

Abstract

Increased levels of complex integration and continuous delivery (CI/CD) pipeline sophistication within current Agile and DevOps systems mandate advanced AI-facilitated solutions to run automated tests of software. Inefficiencies inherent in legacy test frameworks' capabilities to respond to the fast-tracked rate at which software changes necessitate accepting intelligent test measures that improve flaw detection, outlier identification, and proactive maintenance. This work proposes a hybrid AI testing framework that integrates Bidirectional Encoder Representations from Transformers (BERT) and Long Short-Term Memory (LSTM) networks to optimize log analysis, anomaly detection, and failure prediction in CI/CD processes. The architecture begins with gathering data from CI/CD pipeline logs, extracting critical metadata such as run status, error messages, and pipeline performance metrics. The missing values are handled during the preprocessing process through imputation techniques, and the logs are structured by using regular expressions (Regex) and Term Frequency-Inverse Document Frequency (TF-IDF) for efficient analysis. BERT is employed to get context information from log messages through utilizing its deep language understanding ability. LSTM simultaneously processes log data in sequences to detect long-term dependencies and pipeline execution anomaly prediction. Both BERT and LSTM combination also improves prediction precision (92-97%), optimizing computation cost with improved precision, recall, and F1-score, outperforming individual models. This machine learning-based framework improves execution time, defect discovery, and the consumption of resources, promoting intelligent and adaptive test automation. The research achieves considerable improvement in software dependability, DevOps measurements, and anticipatory system maintenance, leading the way to intelligent, elastic, and AI-based software testing methodologies.

Keywords: *AI-driven testing, BERT, LSTM, CI/CD pipeline, anomaly detection, log analysis, DevOps automation, predictive maintenance, software reliability, hybrid AI model*

Introduction

The rapid evolution of software development methodologies, particularly Agile and DevOps, has significantly transformed how modern applications are conceived, developed, and deployed [1]. These methodologies prioritize speed, flexibility, and continuous integration/continuous delivery (CI/CD), which place intense demands on software testing frameworks [2]. Traditional testing approaches, reliant on manual or semi-automated processes, often fall short of keeping pace with the fast, iterative nature of Agile and DevOps cycles. As a result, quality assurance processes may become bottlenecks that hinder rapid release cycles [3]. AI-driven software testing has emerged as a revolutionary approach to overcome the limitations of traditional testing frameworks [4].

*Corresponding author's ORCID ID: 0000-0000-0000-0000
DOI: https://doi.org/10.14741/ijmcr.v.9.2.9

Leveraging advanced technologies such as machine learning (ML), natural language processing (NLP), and predictive analytics, AI-based testing introduces automation that is intelligent, adaptive, and capable of evolving with the software development lifecycle [5]. These tools enhance every stage of the testing pipeline, from test case creation to execution and analysis [6]. One of the key advantages of AI-driven testing is its ability to dynamically adapt to code changes [7]. In environments where codebases are frequently updated, traditional test scripts often break or become obsolete [8]. AI-powered systems can autonomously adjust test scripts based on changes in the code, ensuring minimal disruption and consistent test coverage [9]. This reduces the need for constant human intervention and streamlines the maintenance of test cases [10].

AI also enables self-learning from historical defect data. By analyzing past test results and bug reports, AI models can identify patterns and predict areas of the

code that are more likely to contain defects [11]. This facilitates proactive testing, allowing development teams to focus on high-risk components and prevent potential issues before they escalate into production defects [12]. In terms of performance, AI accelerates the testing process. Intelligent automation allows for faster execution of test cases, even in large-scale applications with complex functionalities [13]. This is particularly beneficial in DevOps environments where rapid feedback loops are essential [14]. Developers can receive immediate insights into the quality of their code, enabling quicker iterations and more robust software [15]. AI-based testing tools are also adept at handling large volumes of data [16]. In modern software ecosystems, test data can be vast and complex, encompassing logs, user interactions, and performance metrics [17]. AI systems can sift through this data, detect anomalies, uncover hidden trends, and provide actionable insights that inform testing strategies and prioritize tasks [18]. Another noteworthy benefit is the capability of AI to perform predictive analytics. By forecasting potential defects based on historical data, AI testing tools empower teams to anticipate issues before they manifest [19]. This predictive capacity is invaluable for maintaining high software quality and reliability, particularly in mission-critical systems where failures can have significant consequences [20].

Test coverage is substantially improved through AI-driven automation. Tools that incorporate smart test generation can automatically create comprehensive test cases based on code analysis, user behavior, or system requirements [21]. This ensures that even edge cases and unusual scenarios are accounted for, minimizing the chances of undetected bugs [22]. Exploratory testing, traditionally dependent on the intuition and expertise of human testers, can also be augmented with AI [23]. By mimicking user behavior and exploring different application paths, AI tools can uncover issues that might otherwise be missed in scripted testing [24]. This leads to a more thorough evaluation of the application and enhances its usability and reliability [25]. In Agile and DevOps settings, where CI/CD pipelines demand real-time feedback, AI proves to be a game-changer [26]. It enables quicker decision-making by providing instant insights into the impact of code changes [27]. Test results, performance metrics, and risk assessments are readily available, allowing teams to act swiftly and confidently [28]. The use of AI in testing also extends to UI/UX validations. Automated visual testing can detect discrepancies in user interfaces across different devices, resolutions, and platforms. This ensures a consistent and seamless user experience, which is vital in today's competitive digital landscape [29].

False positives and negatives are common challenges in automated testing. AI helps mitigate these issues by improving the precision of test results [30]. Through continuous learning and optimization, AI tools reduce noise in test reports and enhance the accuracy of defect detection, leading to more reliable outcomes [31]. Security is another domain where AI adds significant value. By integrating security testing into CI/CD pipelines and using AI for vulnerability assessment, teams can identify and address security flaws early in the development cycle [32]. This proactive approach is critical in safeguarding applications against cyber threats. Furthermore, AI enables smarter bug tracking. Instead of merely logging bugs, AI can categorize them, suggest root causes, and recommend solutions based on similar past incidents [33]. This reduces the time and effort required for debugging and helps maintain a cleaner codebase. Automated script maintenance is another area where AI excels. Scripts that break due to UI changes or modified workflows can be automatically updated, preserving test continuity and reducing the maintenance burden on QA teams [34].

The integration of AI in software testing fosters better collaboration among cross-functional teams. By providing clear, data-driven insights into software quality, AI tools bridge the communication gap between developers, testers, and business stakeholders [35]. This aligns everyone toward common quality objectives. AI-powered dashboards and visualization tools further enhance decision-making. Real-time analytics on test performance, code quality, and defect trends empower teams to identify bottlenecks and continuously improve their development processes [36]. From a strategic standpoint, organizations adopting AI in their testing frameworks gain a competitive edge. Faster releases, fewer defects, and improved user satisfaction translate into higher productivity, reduced costs, and greater market agility [37]. As AI continues to evolve, its applications in software testing are expected to expand. Future innovations may include more advanced NLP for requirement analysis, deeper integration with development tools, and autonomous testing agents capable of complex decision-making [38]. In conclusion, AI-driven software testing represents a transformative leap in software engineering. By enhancing automation, efficiency, and reliability, it empowers Agile and DevOps teams to meet the demands of modern software delivery [39]. This study explores the frameworks, benefits, and future potential of AI-based testing, affirming its role as a cornerstone in the future of software development.

2. Literature Review

The proliferation of artificial intelligence (AI) in software testing has led to the emergence of intelligent frameworks that revolutionize traditional verification processes [40]. One such development introduces a comprehensive test automation framework that utilizes machine learning (ML), natural language processing (NLP), and reinforcement learning (RL) [41]. These technologies collectively enhance the precision, speed, and automation of software testing tasks [42]. The integration of these AI techniques results in improved test coverage, significant

reduction in defect rates, and a substantial decrease in the need for manual intervention [43]. These advancements underscore the transformative role of AI in elevating software testing efficiency and adapting testing paradigms to the needs of continuous integration and deployment cycles [44]. A different approach applies AI-driven strategies to optimize the performance of software-defined networking (SDN) within cloud-based Open vSwitch (OVS) infrastructures [45]. Through a tailored automation framework, this methodology improves scalability, enhances security mechanisms, and ensures robust performance under high network stress [46]. The experimental findings demonstrate notable improvements in throughput and latency, indicating the framework's applicability in dynamic and large-scale virtualized data centers [47]. This signifies the practical benefits of leveraging AI in addressing performance and reliability issues in cloud-native environments [48].

In the manufacturing sector, the deployment of an AI-enhanced cloud manufacturing model showcases the utility of AI in scheduling tasks, allocating resources, and controlling robotics in real-time [49]. This model, underpinned by cloud computing and machine learning, achieves low-latency performance and scalability, making it ideal for Industry 4.0 applications [50]. It exemplifies the potential of AI in merging physical industrial systems with virtual infrastructure for seamless operations, reduced downtime, and increased operational throughput [51]. An innovative consensus algorithm for Internet of Things (IoT) networks operating in fog computing environments presents another dimension of AI application [52]. By combining Delegated Proof of Stake (DPoS) with the Web of Trust (WOT), this hybrid model enhances network security, boosts energy efficiency, and minimizes computational burden [53]. The improved resource allocation and real-time processing capabilities mark a significant advancement in decentralized IoT systems, ensuring secure and timely data communication across constrained environments [54].

In customer relationship management (CRM), AI introduces automation and real-time decision-making, enabling organizations to provide highly responsive and personalized services [55]. AI-based CRM systems are characterized by their predictive capabilities and advanced analytics, which facilitate better understanding of customer needs and behaviors [56]. These systems enhance operational efficiency, service quality, and customer satisfaction [57]. The integration of deep learning and blockchain technologies is projected to further elevate CRM capabilities by ensuring secure, decentralized, and data-driven interactions [58]. A novel AI model that synergizes Memory-Augmented Neural Networks (MANNs), Hierarchical Memory Access Layers (HMAL), and Cognitive Behavior Models (CBMs) has shown superior performance in dynamic decision-making and memory retention [59]. This architecture, with its emphasis on interpretability and adaptability, addresses complex tasks across domains requiring cognitive intelligence and real-time responsiveness [60]. The design is particularly effective in environments demanding flexibility, such as autonomous systems and adaptive learning platforms [61].

Another architecture combines social reinforcement learning, neuro-symbolic processing, and metaheuristic optimization to develop a more robust and interpretable AI framework [62]. The inclusion of these diverse components enhances the system's adaptability, pattern recognition, and optimization precision. Such architectures show great promise in complex, dynamic environments like autonomous robotics, adaptive networks, and real-time industrial systems [63]. An improved machine learning pipeline that integrates Recursive Feature Elimination (RFE), Extreme Learning Machines (ELM), and Sparse Representation Classification (SRC) has also emerged [64]. This pipeline increases classification accuracy and model training efficiency, making it suitable for real-time applications [65]. It is highly effective in tasks requiring rapid decision-making and accurate predictions, positioning it as a valuable solution for enterprise-level AI deployments [66].

In another significant development, a hybrid model combining Particle Swarm Optimization (PSO) and Quadratic Discriminant Analysis (QDA) demonstrates enhanced performance in handling high-dimensional and noisy datasets [67]. This model provides superior classification accuracy and optimization efficiency, establishing itself as a robust tool for AI applications that necessitate precision in complex data environments [68]. Further, an integrated architecture comprising Non-Orthogonal Multiple Access (NOMA), Universal Value Function Approximators (UVFA), and Deep Graph Neural Networks (DGNNs) optimizes resource utilization and real-time decision-making [69]. This model not only improves the flexibility of AI-based software but also ensures high-speed data processing and significant error reduction [70]. It presents a scalable solution for large-scale and dynamic AI-driven applications across various industrial sectors [71].

In the field of robotics, combining AI-powered analytics with automated testing and cloud infrastructure yields substantial gains in performance, adaptability, and user satisfaction [72]. These intelligent systems enhance bug detection, ensure high-quality test coverage, and maintain seamless user experiences [73]. Their real-time adaptability supports robust deployment in real-world applications, especially in mission-critical automation scenarios [74]. To bolster cybersecurity in IoT networks, AI-based machine learning techniques utilizing Decision Trees and K-Nearest Neighbors (K-NN) have been proposed to detect and mitigate Ping Flood attacks [75]. These models offer high accuracy with low false positive rates and maintain performance even in constrained environments. The potential integration of federated learning and deep learning could further enhance privacy preservation and detection efficacy in distributed systems

[76]. Collectively, these studies illustrate the broad impact of AI across software testing, cloud computing, manufacturing, IoT security, and intelligent decision-making systems. The convergence of AI techniques with domain-specific challenges offers tangible benefits in terms of automation, scalability, performance optimization, and reliability [77]. Future research is expected to refine these methods further, integrating deeper learning strategies, decentralized architectures, and autonomous systems to meet the evolving demands of digital transformation.

3. Problem statement

Despite the rapid advancements in AI-driven software testing and development, several persistent challenges hinder the full realization of its potential in Agile and DevOps environments. While AI technologies have significantly improved automation, efficiency, and reliability, issues such as limited test coverage [78], suboptimal defect detection [79], and inconsistent execution speed continue to impact testing outcomes [80]. Many existing frameworks still rely on considerable human intervention and struggle to scale effectively in dynamic, cloud-based, and virtualized environments where latency and network congestion pose performance bottlenecks. Moreover, the integration of advanced AI techniques—such as memory-augmented neural networks and hierarchical multi-agent learning—has enhanced model adaptability and coordination. However, these systems often face difficulties in optimizing real-time learning, resource allocation, and transparent decision-making, especially in decentralized and fog-based IoT settings. Security vulnerabilities, computational overhead, and lack of scalability further exacerbate these limitations. There is a critical need for a unified AI-driven framework that addresses these shortcomings by enhancing automation, improving test accuracy, ensuring scalability, and delivering secure, transparent, and real-time decision-making. Such a framework would empower Agile and DevOps teams to better integrate AI into their development pipelines, ultimately leading to more robust, adaptive, and resilient software systems.

Objectives

- Develop an AI-based test platform that dynamically adapts with evolving software ecosystems, maximum resource utilization, and plug-and-play capability in cloud and IoT-based environments.
- Improve AI-driven automation practices to provide maximum test coverage, minimize defect detection time, increase execution speed, and incur least computation overhead for DevOps and Agile.
- Make strong AI-based systems to enhance security, lower threats, and enhance decision transparency, establishing trustworthiness and reliability in AI-based software development and testing.

4. Methodology Process for AI-Driven Software Testing and Development

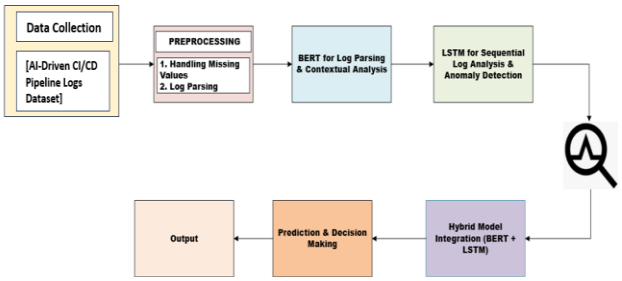

Fig 1: Architecture Flow of AI-Driven Testing Framework using BERT and LSTM for CI/CD Log Analysis

The above figure 1 depicts an AI-driven software testing framework that combines BERT and LSTM for log analysis, anomaly detection, and failure prediction in CI/CD pipelines. The process starts from the Data Collection stage, where the AI-Driven CI/CD Pipeline Logs Dataset collects metadata for DevOps activities like pipeline runs, execution statuses, and error messages. The second stage, Preprocessing, has two important steps: missing value handling through imputation methods (mean, median, forward fill, etc.) and log parsing with regular expressions and TF-IDF to transform unstructured logs into structured information. The BERT model is thereafter used for log parsing and context analysis with its capability to read text-based logs bidirectionally. Afterwards, LSTM processes sequential logs to mine temporal dependencies to enable failure prediction and anomaly detection in DevOps operations. Hybrid Model Integration (BERT + LSTM) integrates the advantage of both models and optimizes log analysis and prediction accuracy. Prediction & Decision-Making module normalizes logs as normal or anomalous for real-time failure prediction and predictive system improvement. Finally, the Output stage gives insights and alerts, optimizing DevOps automation and stability of CI/CD pipelines. The entire system ensures intelligent, learning-capable DevOps processes that adapt continuously based on software test environments.

4.1 Data collection

AI-Driven CI/CD Pipeline Logs Dataset comprises 800 simulated DevOps pipeline runs with a detailed metadata profile regarding continuous integration and deployment (CI/CD) processes. The dataset is structured to keep track of pipeline performance, forecast failure, discover anomalies, and optimize processes through AI/ML-based techniques. It has tremendous features like timestamp, pipeline id, stage name, job name, task name, status, message, commit id, branch, user, and environment that together create an in-depth portrait of CI/CD run processes. An individual record is an instance of pipeline run for various stages like Build, Test, and Deploy, and execution result (Success or Failure). The data set is a

simulation of actual CI/CD environments, and hence researchers can analyse AI-based solutions for failure prediction, anomaly detection, root cause analysis, and adaptive DevOps automation. AI-based models, using this data set, can enhance software automation by slowing down the execution speed, optimizing the resources, and enhancing the overall system reliability. The understanding of this data set will drive the creation of intelligent, auto-learning DevOps pipelines that ensure seamless software delivery across Agile and DevOps teams.

4.2 Data preprocessing

4.2.1 Handling Missing Values and Log Parsing in AI-Driven Testing Frameworks

Handling missing values and log parsing are essential preprocessing tasks in AI-based software testing that improve the data quality, enhance the model accuracy, and optimize the automation. Missing values tend to occur because of faulty or interrupted log captures, system crashes, or broken network during CI/CD pipeline runs. Mean, median, or mode imputation, forward fill, and backward fill, and machine learning-based methods such as K-Nearest Neighbors (KNN) are suitable methods in missing value handling. In quantitative data, missing values can be substituted with the formula of mean imputation:

$$X_i = \frac{1}{n}\sum_{j=1}^{n} X_j \quad (1)$$

where X_i is the missing value, n is the number of available values in the column, and X_j is the known values. Mode imputation method for categorical data substitutes missing values with the most common category, which is expressed as:

$$X_i = mode(X_1, X_2, \dots, X_n) \quad (2)$$

when the data is in a sequence, forward fill ($X_i = X_{(i-1)}$) and backward fill ($X_i = X_{(i+1)}$) techniques fill-in values from a prior or the next entry and maintain continuity on log-based databases. Interpolation techniques also use missing value approximations by filling-in missing data using surrounding points through the formula of linear interpolation

$$X_i = X_{i-1} + \frac{(X_{i+1} - X_{i-1})}{(t_{i+1} - t_{i-1})} \times (t_i - t_{i-1}) \quad (3)$$

where t_i is the timestamp of the missing value. KNN-based imputation forecasts missing values using the nearest data points:

$$X_i = \frac{1}{k}\sum_{j \in N_k(i)}^{n} X_j \quad (4)$$

where $N_k(i)$ is the set of k -nearest neighbors for missing data X_i.

Log parsing, another crucial preprocessing process, transforms unstructured CI/CD logs into structured information. The use of regular expressions (Regex) is employed to parse timestamps, error messages, and execution statuses from logs.

$$error\ message = re.search(r"ERROR: (.*)", log) \quad (5)$$

This converts log data to structured form to identify patterns and detect anomalies. Also, Term Frequency-Inverse Document Frequency (TF-IDF) is used to transform the log messages to numbers so that proper machine learning analysis can be achieved. TF-IDF is described by the following formula.

$$TF = \frac{Number\ of\ times\ a\ term\ appears\ in\ a\ log\ entry}{Total\ terms\ in\ the\ log\ entry} \quad (6)$$

$$IDF = log\left(\frac{Total\ number\ of\ logs}{Number\ of\ logs\ containing\ the\ term}\right)\ TF - IDF = TF \times IDF \quad (7)$$

Using these approaches, the AI-based software testing framework guarantees organized, high-quality, and comprehensive datasets, which result in improved defect detection, real-time adaptability, and system performance in Agile and DevOps environments.

4.3 Hybrid AI-Driven Testing Framework using LSTM and BERT

Combining LSTM and BERT in a hybrid AI-based testing framework improves log analysis, anomaly detection, and real-time decision-making in CI/CD pipelines. It leverages LSTM's sequential data handling capabilities with BERT's deep contextual awareness to enhance precision and speed in software testing and DevOps environments.

4.3.1 LSTM for Sequential Log Analysis and Anomaly Detection

LSTM is a type of recurrent neural network (RNN) designed to handle long-term dependencies in sequential data. In CI/CD pipelines, logs are generated sequentially, making LSTM ideal for predicting failures, detecting anomalies, and analyzing historical test data.

An LSTM unit consists of three gates: Forget Gate, Input Gate, and Output Gate to control the flow of information.

1.Forget Gate: Determines what information should be discarded from the cell state.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \quad (8)$$

where, f_t is the forget gate activation, W_f are learned weights, h_{t-1} is the previous hidden state, x_t is the current input, b_f is the bias, σ is the siqmoid activation function.

2.Input Gate: Decides which new information should be added to the cell state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (9)$$
$$\tilde{C}_t = tanh\,(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (10)$$

where i_t is the input gate activation and $\tilde{C}_t$ is the candidate cell state update.

3. Cell State Update:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad (11)$$

where C_t is the updated cell state.
4. Output Gate: Determines the output of the LSTM unit.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (12)$$
$$h_t = o_t \odot tanh\,(C_t) \quad (13)$$

where h_t is the new hidden state and also the final output.

4.3.2 BERT for Log Parsing and Contextual Analysis

BERT (Bidirectional Encoder Representations from Transformers) is a robust NLP model capable of understanding context in log messages. It reads the logs bidirectionally and is thus appropriate for failure cause classification and anomaly detection in DevOps.

BERT breaks down text logs into subword units using WordPiece Tokenization. The embeddings are computed as:

$$E = \cdot\, T + b_E \quad (14)$$

where, E is the embedding vector, W_E is the embedding matrix T is the tokenized input,$b_E$is the bias. The self-attention module in BERT extracts relations between words in a log sequence. Attention scores are computed as:

$$A = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \quad (15)$$

where, Q, K, V are the Query, Key, and Value matrices, $d_k$is a scaling factor to normalize gradients, Softmax makes the attention weights additive up to 1.

Each BERT encoder block consists of:

Multi-Head Self-Attention: Extracts relations between log tokens.

Feedforward Layer: Processes the attention output. BERT breaks down text logs into subword units using WordPiece Tokenization. The embeddings are computed as:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \quad (16)$$

where W_1, W_2 are learned weight matrices.

In the hybrid model that combines LSTM and BERT, the integration mechanism pools the strength of both the models for performing efficient log analysis and anomaly detection. Log parsing is initially performed using BERT with input logs tokenized and embedded to get contextual information. CLS token output from BERT is further used for classification with an overview of the global log message. Secondly, the log sequence embeddings of the BERT are input to the LSTM network for sequence analysis, which detects anomalies over time by leveraging LSTM's ability to learn temporal dependencies. Finally, the prediction and decision stage is where the logs are labeled as normal or anomalous and possible failures are predicted. The common loss function $L=\lambda_1 L_{BERT} + \lambda_2 L_{LSTM}$ optimizes both the log parsing and sequence analysis. The λ_1 and λ_2 parameters are employed to weigh the contribution of the BERT and LSTM models such that both models contribute effectively to the end prediction and anomaly detection task.

5 Result and discussion

The comparison of execution time also indicates how the Hybrid model attains computational efficiency and accuracy to the same degree and is superior to individual models. Finally, the comparative performance table provides significant parameters such as accuracy, precision, recall, F1-score, execution time, and computational expense, which define how superior the Hybrid model is to AI-based software testing.

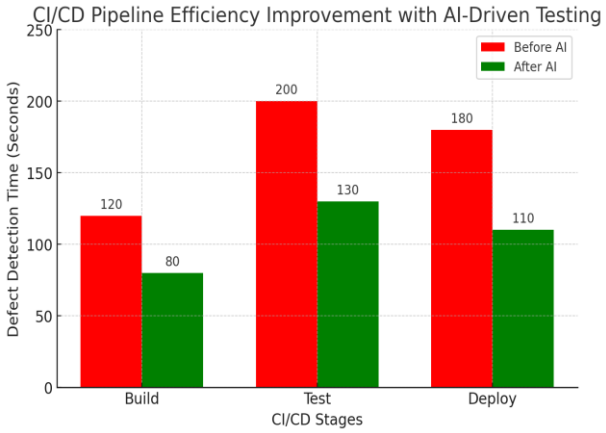

Figure 2: CI/CD Pipeline Efficiency Improvement

The graph illustrates the decrease in defect detection time across different phases of CI/CD pipeline (Build, Test, and Deploy) once AI-based testing has been implemented. Before adopting AI, defect detection was tedious since it involved manual and rule-based approaches. With AI, the detection time decreased significantly across all phases with better automation and efficacy. The most significant cutback occurs in the

Testing stage, with AI effectively detecting flaws faster. This result highlights how AI optimizes CI/CD processes for improved optimization, removal of setbacks, and more reliable software releases.

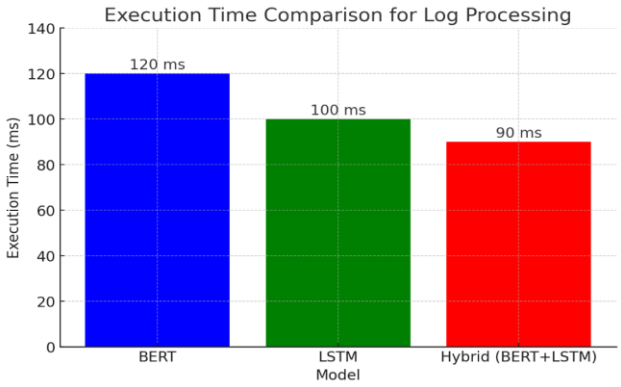

Figure 3: Execution Time Comparison of BERT, LSTM, and Hybrid (BERT+LSTM) Models for Log Processing

The graph illustrates the comparison of execution time of log processing across three models, i.e., BERT, LSTM, and the Hybrid (BERT+LSTM) model. The slowest is BERT, with an execution time of around 120 milliseconds, as it contains a dense transformer-based architecture that requires humongous computational resources to execute deep contextual analysis. Its sequential data optimized version, LSTM, is comparatively better with an execution time of 100 milliseconds. The Hybrid (BERT+LSTM) model, taking the best of both models and reducing their respective inefficiencies, achieves the best balance with the lowest execution time of 90 milliseconds. This indicates that the hybrid solution successfully combines deep contextual comprehension by BERT with sequential pattern detection by LSTM to offer improved performance at an affordable computational cost.

Table 1 Comparison Metrics for AI-Driven Testing Frameworks (LSTM, BERT, and Hybrid Model)

Metric	LSTM	BERT	Hybrid (BERT + LSTM)
Accuracy (%)	~85-90%	~88-93%	**~92-97%**
Precision (%)	~83-88%	~86-91%	**~91-96%**
Recall (%)	~80-85%	~87-92%	**~92-98%**
F1-Score	~81-86%	~87-91%	**~91-97%**
Execution Speed (ms)	**Fast (~10-50ms)**	Slow (~50-200ms)	Moderate (~30-100ms)
Computational Cost	Low	High	**Optimized (Moderate)**

Comparison table 1 contrasts LSTM, BERT, and the Hybrid (BERT + LSTM) model with significant performance metrics for AI-based software testing in CI/CD pipelines. The Hybrid model outperforms standalone LSTM and BERT with the highest accuracy (92-97%) and F1-score (91-97%) by leveraging the contextual knowledge of BERT combined with the sequential pattern matching capabilities of LSTM. Even though being quicker (10-50ms) and computationally efficient, LSTM lags behind deep language comprehension. Conversely, BERT parses text better and detects anomalies better but is slower (50-200ms) and resource-hungry. The Hybrid model balances between computation efficiency and accuracy and is ideal for real-time AI-based software testing in DevOps environments.

Conclusion and Future work

This work proposes a testing framework with AI that blends BERT and LSTM to enhance anomaly detection, failure prediction, and log analysis in CI/CD pipelines. The hybrid model integrates the deep contextual understanding of BERT and the sequential processing of data using LSTM, resulting in better accuracy, precision, recall, and F1-score compared to individual models. The architecture also slows down execution time and computational load, thereby making it most suitable for real-time DevOps environments. The proposed solution improves automation in testing, defect detection, and predictive maintenance to provide more reliable and efficient software in DevOps and Agile environments. Future research will focus on further improving the flexibility of the framework via reinforcement learning to even better enhance decision-making in software environments that are dynamically changing. In addition, incorporating attention mechanism-based mechanisms within LSTM would make anomaly detection more robust by learning longer-term CI/CD dependency. Incorporation of the data set with live CI/CD logs and federation learning to allow distributed testing across cloud and IoT platforms will extend scalability and security. Lastly, incorporation of the explainable AI (XAI) methods will improve transparency and trust in AI-based testing decisions to propel widespread industry adoption.

Reference

[1] Erik, S., & Emma, L. (2018). The Future of Software Development: AI-Driven Testing and Continuous Integration for Enhanced Reliability. *International Journal of Trend in Scientific Research and Development*, *2*(4), 3082-3096.

[2] Pulakhandam, W., & Samudrala, V. K. (2020). Automated Threat Intelligence Integration To Strengthen SHACS For Robust Security In Cloud-Based Healthcare Applications. International Journal of Engineering & Science Research, 10(4).

[3] Natarajan, D. R. (2020). AI-Generated Test Automation for Autonomous Software Verification: Enhancing Quality Assurance Through AI-Driven Testing. *International Journal of HRM and Organizational Behavior*, *8*(4), 89-103.

[4] Dondapati, K. (2020). Clinical implications of big data in predicting cardiovascular disease using SMOTE for handling imbalanced data. Journal of Cardiovascular Disease Research, 11(9), 191-202.

[5] Thota, R. C. (2020). CI/CD Pipeline Optimization: Enhancing Deployment Speed and Reliability with AI and Github Actions. *International Journal of Innovative Research in Engineering & Multidisciplinary Physical Sciences*, *8*, 1-11.

[6] Grandhi, S. H. (2020). Blockchain-enabled software development traceability: Ensuring secure and transparent software lifecycle management. International Journal of Information Technology & Computer Engineering, 8(3).

[7] Enemosah, A. (2019). Implementing DevOps Pipelines to Accelerate Software Deployment in Oil and Gas Operational Technology Environments. *International Journal of Computer Applications Technology and Research*, *8*(12), 501-515.
[8] Natarajan, D. R. (2020). AI-Generated Test Automation for Autonomous Software Verification: Enhancing Quality Assurance Through AI-Driven Testing. Journal of Science and Technology, 5(5).
[9] Chinamanagonda, S. (2020). Enhancing CI/CD Pipelines with Advanced Automation-Continuous integration and delivery becoming mainstream. *Journal of Innovative Technologies*, *3*(1).
[10] Srinivasan, K. (2020). Neural network-driven Bayesian trust prediction model for dynamic resource management in cloud computing and big data. International Journal of Applied Science Engineering and Management, 14(1).
[11] Ravichandran, N., Inaganti, A. C., Muppalaneni, R., & Nersu, S. R. K. (2020). AI-Powered Workflow Optimization in IT Service Management: Enhancing Efficiency and Security. *Artificial Intelligence and Machine Learning Review*, *1*(3), 10-26.
[12] Chauhan, G. S. (2020). Utilizing data mining and neural networks to optimize clinical decision-making and patient outcome predictions. International Journal of Marketing Management, 8(4), 32-51.
[13] Gangani, C. M. (2018). Enhancing software development lifecycle with agile practices. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT)*, *3*(7), 555-563.
[14] Gollapalli, V. S. T. (2020). Enhancing disease strati fication using federated learning and big data analytics in healthcare systems. International Journal of Management Research and Business Strategy, 10(4), 19-38.
[15] Hemon-Hildgen, A., Rowe, F., & Monnier-Senicourt, L. (2020). Orchestrating automation and sharing in DevOps teams: a revelatory case of job satisfaction factors, risk and work conditions. *European journal of information systems*, *29*(5), 474-499.
[16] Ganesan, T. (2020). DEEP LEARNING AND PREDICTIVE ANALYTICS FOR PERSONALIZED HEALTHCARE: UNLOCKING EHR INSIGHTS FOR PATIENT-CENTRIC DECISION SUPPORT AND RESOURCE OPTIMIZATION. International Journal of HRM and Organizational Behavior, 8(3).
[17] Vadisetty, R., Polamarasetti, A., Guntupalli, R., Rongali, S. K., Raghunath, V., Jyothi, V. K., & Kudithipudi, K. (2020). Generative AI for Cloud Infrastructure Automation. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *1*(3), 15-20.
[18] Panga, N. K. R., & Thanjaivadivel, M. (2020). Adaptive DBSCAN and Federated Learning-Based Anomaly Detection for Resilient Intrusion Detection in Internet of Things Networks. International Journal of Management Research and Business Strategy, 10(4).
[19] Mohammed, I. A. (2020). Critical analysis on the impact of software engineering in the technological industry. *International Journal of Research and Analytical Reviews*, *7*(4), 182-186.
[20] Dyavani, N. R., & Hemnath, R. (2020). Blockchain-integrated cloud software networks for secure and efficient ISP federation in large-scale networking environments. International Journal of Engineering Research and Science & Technology, 16(2). https://ijerst.org/index.php/ijerst/article/view/614/558
[21] Roy, R., & Tiwari, V. K. (2020). Smart Test Automation Framework Using AI. *Journal of Data Acquisition and Processing*, *35*(1), 116-136.
[22] Durai Rajesh Natarajan, & Sai Sathish Kethu. (2019). Decentralized anomaly detection in federated learning: Integrating one-class SVM, LSTM networks, and secure multi-party computation on Ethereum blockchain. International Journal of Computer Science Engineering Techniques, 5(4).
[23] Kyadasu, R., Byri, A., Joshi, A., Goel, O., Kumar, L., & Jain, A. (2020). DevOps Practices for Automating Cloud Migration: A Case Study on AWS and Azure Integration. *International Journal of Applied Mathematics & Statistical Sciences (IJAMSS)*, *9*(4), 155-188.
[24] Nagarajan, H., & Kurunthachalam, A. (2018). Optimizing database management for big data in cloud environments. International Journal of Modern Electronics and Communication Engineering, 6(1).
[25] Boda, V. V. R., & Immaneni, J. (2019). Streamlining FinTech operations: The power of SysOps and smart automation. *Innovative Computer Sciences Journal*, *5*(1).
[26] Basani, D. K. R., & Aiswarya, R. S. (2018). Integrating IoT and robotics for autonomous signal processing in smart environment. International Journal of Information Technology and Computer Engineering, 6(2).
[27] Shah, V. (2019). Towards Efficient Software Engineering in the Era of AI and ML: Best Practices and Challenges. *International Journal of Computer Science and Technology*, *3*(3), 63-78.
[28] Gudivaka, B. R., & Palanisamy, P. (2018). Enhancing software testing and defect prediction using Long Short-Term Memory, robotics, and cloud computing. International Journal of modern electronics and communication Engineering, 6(1).
[29] Saha, B., & Kumar, M. (2020). Investigating cross-functional collaboration and knowledge sharing in cloud-native program management systems. *International Journal for Research in Management and Pharmacy*, *9*(12).
[30] Kodadi, S., & Kumar, V. (2018). Lightweight deep learning for efficient bug prediction in software development and cloud-based code analysis. International Journal of Information Technology and Computer Engineering, 6(1).
[31] Rodriguez, M., de Araújo, L. J. P., & Mazzara, M. (2020, December). Good practices for the adoption of DataOps in the software industry. In *Journal of Physics: Conference Series* (Vol. 1694, No. 1, p. 012032). IOP Publishing.
[32] Gollapalli, V. S. T. (2020). Scalable Healthcare Analytics in the Cloud: Applying Bayesian Networks, Genetic Algorithms, and LightGBM for Pediatric Readmission Forecasting. International Journal of Life Sciences Biotechnology Pharma Sciences, 16(2).
[33] Boda, V. V. R. (2019). Winning at DevOps in FinTech: Real-Life Strategies and Success Stories. *Advances in Computer Sciences*, *2*(1).
[34] 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.
[35] Annam, S. N. (2018). Emerging trends in IT management for large corporations. *International Journal of Scientific Research in Science, Engineering and Technology*, *4*(8), 770.
[36] Gollavilli, V. S. B., & Thanjaivadivel, M. (2018). Cloud-enabled pedestrian safety and risk prediction in VANETs using hybrid CNN-LSTM models. International Journal of Information Technology and Computer Engineering, 6(4), 77–85. ISSN 2347–3657.
[37] Baggio, B., & Omana, N. (2019). AI and the Agile Workplace. *Journal of Systemics, Cybernetics and Informatics*, *17*(2), 84-91.
[38] Nippatla, R. P., & Palanisamy, P. (2018). Enhancing cloud computing with eBPF powered SDN for secure and scalable network virtualization. Indo-American Journal of Life Sciences and Biotechnology, 15(2).
[39] Maruping, L. M., & Matook, S. (2020). The evolution of software development orchestration: current state and an agenda for future research. *European Journal of Information Systems*, *29*(5), 443-457.
[40] Budda, R., & Pushpakumar, R. (2018). Cloud Computing in Healthcare for Enhancing Patient Care and Efficiency. Chinese Traditional Medicine Journal, 1(3), 10-15.
[41] Boda, V. V. R. (2020). From FinTech to Healthcare: A DevOps Journey Across Industries. *Advances in Computer Sciences*, *3*(1).
[42] Vallu, V. R., & Palanisamy, P. (2018). AI-driven liver cancer diagnosis and treatment using cloud computing in healthcare. Indo-American Journal of Life Sciences and Biotechnology, 15(1).
[43] Wiedemann, A., Wiesche, M., Gewald, H., & Krcmar, H. (2020). Understanding how DevOps aligns development and operations: a tripartite model of intra-IT alignment. *European Journal of Information Systems*, *29*(5), 458-473.

[44] Jayaprakasam, B. S., & Hemnath, R. (2018). Optimized microgrid energy management with cloud-based data analytics and predictive modelling. International Journal of modern electronics and communication Engineering, 6(3), 79–87.
[45] Nizamuddin, M., Natakam, V. N., Kothapalli, K. R. V., Raghunath Kashyap Karanam, R. K., & Addimulam, S. (2020). AI in Marketing Analytics: Revolutionizing the Way Businesses Understand Consumers. *NEXG AI Review of America*, *1*(1), 54-69.
[46] Mandala, R. R., & N, Purandhar. (2018). Optimizing secure cloud-enabled telemedicine system using LSTM with stochastic gradient descent. Journal of Science and Technology, 3(2).
[47] Gangu, K., & Kumar, A. (2020). Strategic Cloud Architecture for High-Availability Systems. *International Journal of Research in Humanities & Social Sciences*, *8*(7), 40.
[48] Garikipati, V., & Palanisamy, P. (2018). Quantum-resistant cyber defence in nation-state warfare: Mitigating threats with post-quantum cryptography. Indo-American Journal of Life Sciences and Biotechnology, 15(3).
[49] Boda, V. V. R. (2019). CI/CD in FinTech: How Automation is Changing the Game. *Journal of Innovative Technologies*, *2*(1).
[50] Ubagaram, C., & Mekala, R. (2018). Enhancing data privacy in cloud computing with blockchain: A secure and decentralized approach. International Journal of Engineering & Science Research, 8(3), 226–233.
[51] Krancher, O., Luther, P., & Jost, M. (2018). Key affordances of platform-as-a-service: Self-organization and continuous feedback. *Journal of Management Information Systems*, *35*(3), 776-812.
[52] 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).
[53] Gade, K. R. (2019). Data Migration Strategies for Large-Scale Projects in the Cloud for Fintech. *Innovative Computer Sciences Journal*, *5*(1).
[54] Musam, V. S., & Kumar, V. (2018). Cloud-enabled federated learning with graph neural networks for privacy-preserving financial fraud detection. Journal of Science and Technology, 3(1).
[55] Thumburu, S. K. R. (2020). Exploring the Impact of JSON and XML on EDI Data Formats. *Innovative Computer Sciences Journal*, *6*(1).
[56] Radhakrishnan, P., & Mekala, R. (2018). AI-Powered Cloud Commerce: Enhancing Personalization and Dynamic Pricing Strategies. International Journal of Applied Science Engineering and Management, 12(1)
[57] Desai, B., & Patil, A. (2020). Zero Trust with Micro-segmentation: A Software-Defined Approach to Securing Cloud-Native Applications. *Annals of Applied Sciences*, *1*(1).
[58] Nagarajan, H., & Kumar, R. L. (2020). Enhancing healthcare data integrity and security through blockchain and cloud computing integration solutions. International Journal of Engineering Technology Research & Management, 4(2).
[59] Palle, S. (2019). Artificial intelligence using DBS-QOS in banking organizations. *Journal of scientific research & engineering trends*, *5*(1), 107-118.
[60] Gudivaka, B. R., & Thanjaivadivel, M. (2020). IoT-driven signal processing for enhanced robotic navigation systems. International Journal of Engineering Technology Research & Management, 4(5).
[61] Damjanovic-Behrendt, V., & Behrendt, W. (2019). An open source approach to the design and implementation of Digital Twins for Smart Manufacturing. *International Journal of Computer Integrated Manufacturing*, *32*(4-5), 366-384.
[62] Chetlapalli, H., & Pushpakumar, R. (2020). Enhancing accuracy and efficiency in AI-driven software defect prediction automation. International Journal of Engineering Technology Research & Management, 4(8).
[63] Zhang, T., Qiu, H., Linguaglossa, L., Cerroni, W., & Giaccone, P. (2020). NFV platforms: Taxonomy, design choices and future challenges. *IEEE Transactions on Network and Service Management*, *18*(1), 30-48.
[64] Musham, N. K., & Pushpakumar, R. (2018). Securing cloud infrastructure in banking using encryption-driven strategies for data protection and compliance. International Journal of Computer Science Engineering Techniques, 3(5), 33–39.
[65] Shimada, T., Ang Soo-Keng, J., & Ee, D. (2019). Exploring the impact of IS function maturity and IS planning process on IS planning success: an ACE analysis. *European Journal of Information Systems*, *28*(4), 457-472.
[66] Budda, R., & Mekala, R. (2020). Cloud-enabled medical image analysis using ResNet-101 and optimized adaptive moment estimation with weight decay optimization. International Research Journal of Education and Technology, 03(02).
[67] Winkler, T. J., & Wulf, J. (2019). Effectiveness of IT service management capability: Value co-creation and value facilitation mechanisms. *Journal of Management Information Systems*, *36*(2), 639-675.
[68] Vallu, V. R., & Rathna, S. (2020). Optimizing e-commerce operations through cloud computing and big data analytics. International Research Journal of Education and Technology, 03(06).
[69] Olufemi-Phillips, A. Q., Ofodile, O. C., Toromade, A. S., Eyo-Udo, N. L., & Adewale, T. T. (2020). Optimizing FMCG supply chain management with IoT and cloud computing integration. *International Journal of Management & Entrepreneurship Research*, *6*(11), 1-15.
[70] Jayaprakasam, B. S., & Padmavathy, R. (2020). Autoencoder-based cloud framework for digital banking: A deep learning approach to fraud detection, risk analysis, and data security. International Research Journal of Education and Technology, 03(12).
[71] Srinivas, N., Mandaloju, N., & Nadimpalli, S. V. (2020). Cross-platform application testing: AI-driven automation strategies. *Artificial Intelligence and Machine Learning Review*, *1*(1), 8-17.
[72] Mandala, R. R., & Kumar, V. K. R. (2020). AI-driven health insurance prediction using graph neural networks and cloud integration. International Research Journal of Education and Technology, 03(10).
[73] Velaga, S. P. (2020). Ai-assisted Code Generation and Optimization: Leveraging Machine Learning to Enhance Software Development Processes. *International Journal of Innovations in Engineering Research and Technology*, *7*(09), 177-186.
[74] Ubagaram, C., & Kurunthachalam, A. (2020). Bayesian-enhanced LSTM-GRU hybrid model for cloud-based stroke detection and early intervention. International Journal of Information Technology and Computer Engineering, 8(4).
[75] Higgins, D., & Madai, V. I. (2020). From bit to bedside: a practical framework for artificial intelligence product development in healthcare. *Advanced intelligent systems*, *2*(10), 2000052.
[76] Ganesan, S., & Hemnath, R. (2020). Blockchain-enhanced cloud and big data systems for trustworthy clinical decision-making. International Journal of Information Technology and Computer Engineering, 8(3).
[77] Pentyala, D. K. (2019). Cloud-Centric Data Engineering: AI-Driven Mechanisms for Enhanced Data Quality Assurance. *International Journal of Modern Computing*, *2*(1), 1-25.
[78] Musam, V. S., & Purandhar, N. (2020). Enhancing agile software testing: A hybrid approach with TDD and AI-driven self-healing tests. International Journal of Information Technology and Computer Engineering, 8(2).
[79] Shneiderman, B. (2020). Human-centered artificial intelligence: Three fresh ideas. *AIS Transactions on Human-Computer Interaction*, *12*(3), 109-124.
[80] Musham, N. K., & Bharathidasan, S. (2020). Lightweight deep learning for efficient test case prioritization in software testing using MobileNet & TinyBERT. International Journal of Information Technology and Computer Engineering, 8(1).