Journal of Information Systems Engineering and Management
2025, 10(36s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Harnessing Big Data Analytics and Deep Learning for Predictive Bug Finding and Test Automation in Complex Embedded Software Systems

Tayar Yerramsetty[1], Mohammed Moazzam Moinuddin[2], Kishore Ranjan[3], Ian Pranandi[4], V M Gobinath[5], Kalpesh Rasiklal Rakholia[6]

[1]*Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur District, 522502, Andhra Pradesh, India. Email ID: tayaryerramsetty@kluniversity.in*

[2]*Associate Professor, Electronics and Communication Engineering, Maulana Azad National Urdu University Polytechnic Bangalore, India. Email ID: moazzam_95@yahoo.com*

[3]*ACM SIGBED, SEMI, Trumbull, Connecticut, USA. Email ID: kishoreranjan@gmail.com*

[4]*Department of Biochemistry, School of Medicine and Health Sciences, Atma Jaya Catholic University of Indonesia, Jakarta 14440, Indonesia. Email ID: ian.pranandi@atmajaya.ac.id*

[5]*Department of Mechanical, Rajalakshmi Institute of Technology, Chennai, India, Email ID : vmgobinath@gmail.com*

[6]*Assistant Professor, PIET-IT, Parul University-Vadodara, Gujarat, India, Email ID: kraykholiya@gmail.com*

ARTICLE INFO

Received: 18 Dec 2024

Revised: 10 Feb 2025

Accepted: 28 Feb 2025

ABSTRACT

Deep learning and data large analytics form a powerful combination for predictive prediction of bugs in the complicated embedded systems and for automated test. We confirm that such approaches can achieve good accuracy and efficiency in comparison with conventional methods. This work is based on the finding of which is to encourage adoption of AI driven testing strategies that form a foundation for future software reliability and intelligent defect management innovations in safety critical and larger scale embedded applications.

Keywords: Test Automation, Embedded Software, Predictive Bug, Big Data Analytics, Deep Learning.

Introduction

Embedded systems rapidly becoming more complex and ensuring the reliability of software and high rate of testing is increasingly important. Traditional solutions for bug detection and validation are not suitable under these demands. In this research, whether big data analytics and deep learning can be used to revolutionize the deployment and predictive bug detection, as well as the test automation activities is tested. The study shows how intelligent systems can improve efficiency, reduce defects and speed up delivery of embedded software on the basis of machine learning on historical data.

Related Works

Over the recent past, there has been convergence of big data analytics, deep learning, and the embedded software testing. As the embedded systems become more complex and require rapid and reliable software release, intelligently automating and modelling predict world has become indispensable.

Existing literature shows wide methodological and variegated approaches in attempting quality of software, system testing efficiency and precision in detecting bugs especially with the use of machine learning (ML) and deep learning (DL).

Predictive Analytics

In his work, Al Hanash (2023) demonstrates how predictive data analytics is a transformative tool in industrial application particularly for embedded software testing environment. With the use of supervised ML models like – support vector machines, and random forests; gradient boosting; and Multi-Layer Perceptron; organizations can predict failures and defects before they happen.

Journal of Information Systems Engineering and Management
2025, 10(36s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

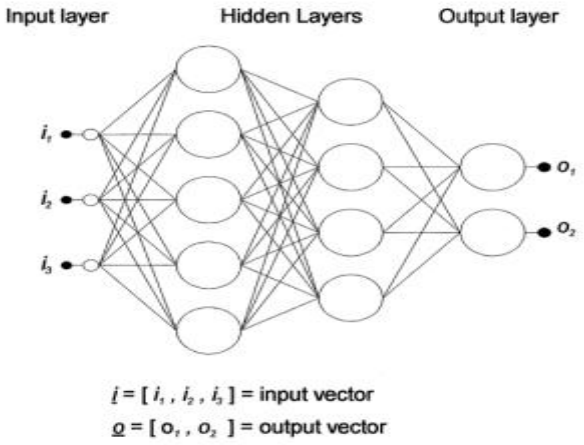


Figure 1 Multi-layer Perceptron (Al Hanash, 2023)

In the second case, such models utilize the structured data and are evaluated with metrics such as recall, precision, F1 score, and ROC AUC to gauge their effectiveness. In addition to supporting predictive maintenance, this approach also provides for reinforcing test systems with data driven intelligence. Integrating Predictive Analytics into Embedded Systems that require Failure prediction in real-time is set up as a precedent.

Bug Characteristics

Islam et al. (2019) stated common defect pattern and bug root causes in popular DL frameworks are investigated. According to their empirical analysis, improper arrangements of parameters and poor design of structures are major causes of bugs in deep learning code. Stages demonstrating particularly high levels of bug proneness show that the model construction and training stages are interestingly.

Additionally, common DL anti pattern such as improper API use or incorrect layer ordering indicate the need for automated validation. They suggest the development of specific debugging tools aimed at the domain of deep learning to tackle errors that are otherwise hard to be spotted manually because DL pipelines are basically abstract.

Automated Testing

Nama (2023) states that ML improves test case generation and defect prediction. ML models analyzing historical data and software metrics can facilitate focusing and reducing the testing to accurately find the regions that most likely cause the bugs. Unlike the manual, the automated test case generation approach has a broader coverage with less manual effort, compared to the traditional test case generation.

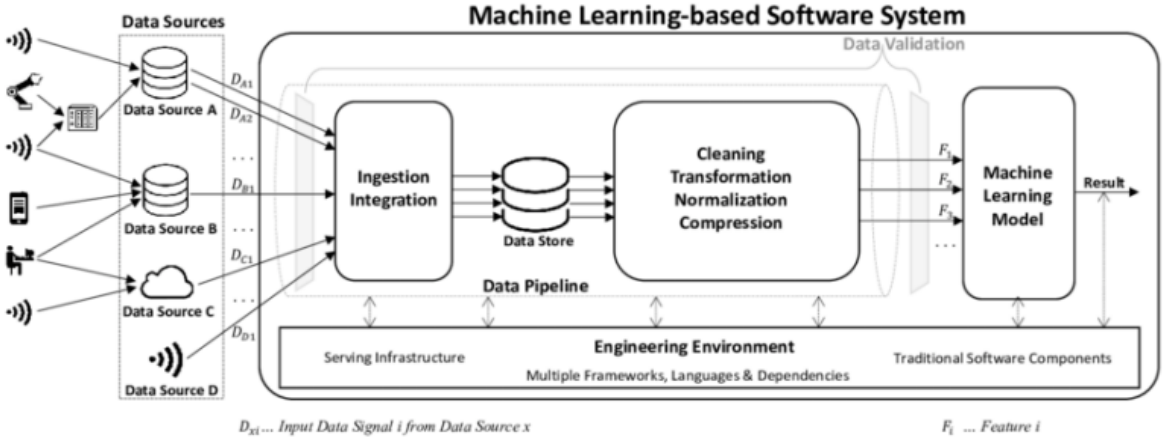


Figure 2 Machine Learning system (Nama, 2023)

Journal of Information Systems Engineering and Management
2025, 10(36s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

This also translates to higher accuracy of identifying hidden defects. Specifically, this is a strong approach in agile and DevOps environments on account of the time-to-market pressures that call for a more efficient testing strategy.

Table 1: Comparative Analysis

ML Technique	Testing Functionality	Strengths	Limitations
Decision Trees	Defect prediction in software system using decision trees is very common.	This approach is easy to read and has a low computational cost, therefore allowing for it to be used indoors as well as outdoors in a variety of testing environments.	Despite that, decision trees can over fit and they tend to struggle in learning patterns of relationships with noisy data.
Random Forest	In the large scale of projects, random forest is commonly used to prioritize test cases.	It achieves high accuracy, effectively reduces overfitting, and is also effective on high dimensional datasets of large size.	As a downside to the random forest models, they are less interpretable than the simple single decision tree, and sometimes require excessive computational resources.
Support Vector Machines (SVM)	They are used to detect faults and classify defective modules using SVMs.	Such models are very good compared with overfitting in high dimensions feature space.	However, SVMs are not easy to tune the parameters in and may not scale as well as require aggressive data reduction with very large datasets.
Neural Networks	To run automated testing systems, both test case generation and optimization are done using neural networks.	One can model non-linear relationships or scale to large amount of data.	A drawback is their dependency on high volumes of training data, and they are not interpretable, therefore the debugging is difficult.
Naïve Bayes	For classifying bugs based on textual features in bug reports, Naïve Bayes classifiers are widely used.	Specifically, they are very fast and efficient in tasks involving natural language processing, in data sets that have textual data.	Naïve Bayes might fail on unbalanced datasets and with a poor performance on the general case if feature independence is assumed, however rarely is it the case in practice.

Comparative Analysis

The meta-analysis of ML based test methods is provided by Ajorloo et al. (2024). Their work shows that while machine learning methods offer very good promise in fault detection and test optimization, existing techniques are generally not generalizable to different types of software systems.

They name the open problems that can hinder the ML models like explainability of ML models and construction of sufficient false positive and false negatives. And their work becomes a roadmap for future research proposing hybrid models and multi objective optimization method as two promising directions in future to achieve higher prediction accuracy and scalability of the real-world testing environments.

Testing Challenges

Specifically, Braiek and Khomh (2020) are interested in the issues pertaining to challenges of testing in ML based applications, especially in safety critical systems. Almost all ML systems are getting wider usage and, hence, the demand for ensuring them reliable is on the rise.

Journal of Information Systems Engineering and Management
2025, 10(36s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

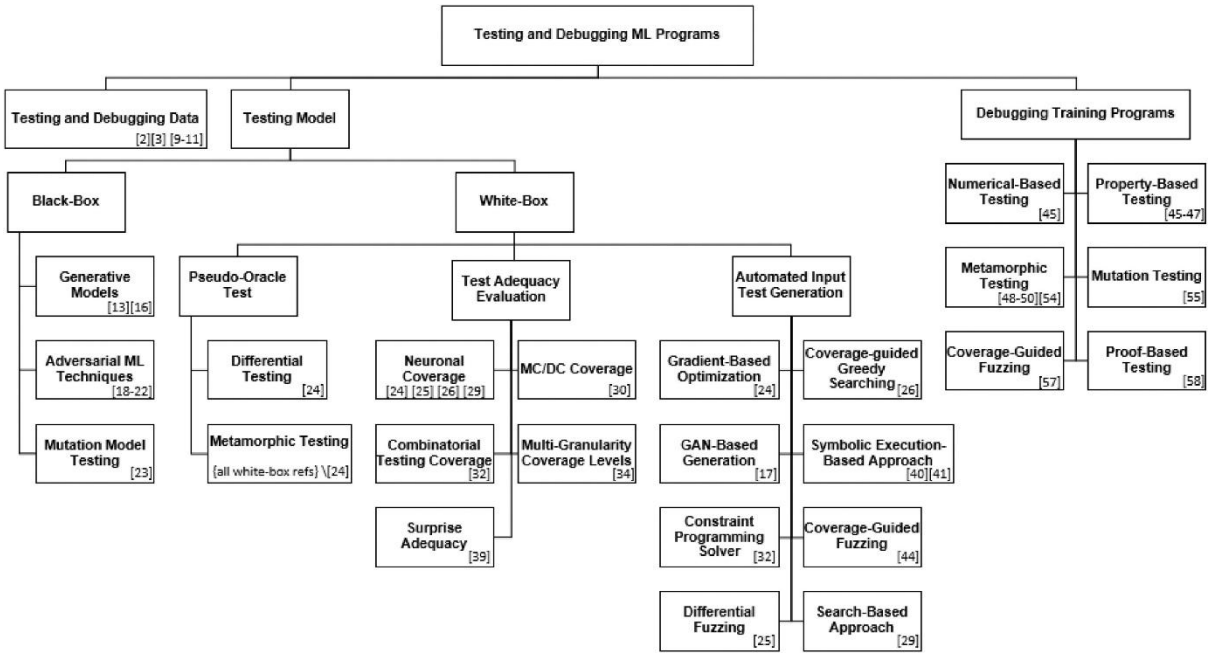


Figure 3 Testing and Debugging (Nraiek and Khomh, 2020)

The ML outputs are non-deterministic, making the adaptation difficult. To support the stochastic behavior of the ML systems, the paper suggests changing software testing strategies and argues to include explainable AI to enhance the transparency during debugging and validation processes.

Foundations of ML

According to Sivaraman (2020), ML serves as a prevalent engine in modern software engineering enabling the automating code analysis, defect detection and test generation. Adaptive learning is supported by ML in the sense in which systems can get better as they learn from past performance.

This is very useful for situations in which manual monitoring is not practical because of the high dimensions of the system. The study argues that predictive modeling and pattern recognition through ML can greatly contribute to improving software reliability and cost reduction for maintenance.

Semantic Bug Detection

In Pradel and Sen (2018), they present a deep learning based semantic bug detection method. It teaches a classifier on correct as well as synthetically generated buggy code snippets, so leading to the good programmer mistakes such as argument swaps or wrong operators.

This model shows high efficiency and accuracy for generalization of DL from artificially created bug instances to actual world issues. Especially it can be integrated into continuous integration (CI) systems providing real time feedback to the developers.

Table 2: Types of Bugs

Bug Type	Description	Detection Technique	Examples
Argument Order Swaps	This is a bug of type here, in cases where function arguments are passed in the incorrect order, almost always by coincidence similar type or naming conventions.	Deep learning models with semantic embedding to the variable and function names are used to detect.	For example, the incorrect call copy(src, dest) instead of the intended copy(dest, src) may be flagged as a potential bug.

Journal of Information Systems Engineering and Management
2025, 10(36s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Incorrect Operators	The bugs mentioned above come from misusing math or logic operator in expressions.	Both deep learning models and classification method are used to learn correct patterns from large code corpora and then detect deviations from the correct patterns.	Using == in place of !=
Wrong Function Calls	This bug type is the case of calling the wrong or semantically incorrect function into a specified context.	The model gets contextual embeddings of identifiers to understand the functional relevance and identify inappropriate function calls.	Using openFile() and readFile()
Type Mismatches	The issue of type mismatch bugs happens when we assign a value that is of an incompatible data type.	Var declaration and assign are analyzed in deep context aware learning.	A typical example is assigning a string to a variable meant to hold an integer in some dynamically typed languages that humanity is likely to compile and run.
Loop Boundary Errors	These bugs stem from bad boundary conditions on iterative construct, i.e, for or while.	Standard loop patterns are recognized and anomalies that suggest off-by-one or range related errors, are detected by neural models trained to detect them.	An example is the W in the above in writing for (i <= n) rather than the correct for (i < n), which would mean having an iteration or an index out of bound error.

Software Quality

While the importance of robust software quality practices in embedded systems is emphasized in the context of aerospace and automotive domain in Fariha et al. (2024). As their systematic literature review shows, there is a growing interest to apply AI techniques in order to improve the requirements engineering and the maintenance processes.

tThey also point out the gaps especially related to automation of verification of nonfunctional requirements and their software maintainability for the long term. However, applied ML when used in such areas is still underexplored, which provides a potential niche for future research to help improve the reliability of the long-term embedded software.

Defect Prediction

Defect prediction is a fundamental task for embedded software development, as pointed out in Thota et al. (2020). This study asserts an ability to identify defect prone modules early from which testing costs can be reduced and resources are allocated effectively. Soft computing techniques (fuzzy logic, neural networks) are used to use of such models able to handle uncertain and imprecise information. These approaches provide a cost-effective approach to traditional testing strategies of embedded systems with constrained environments that are expensive and time consuming to test.

Predictive Modeling

As it is a crucial problem in predictive bug finding, Krishnan et al. (2017) address this problem of data quality in ML models. BoostClean is their framework that detects and fixes domain value violations which often degrade model's accuracy, automatically. BoostClean improves the prediction performance by taking advantage of ensemble learning and deep semantic sense (i.e Word2Vec). However, this work emphasizes the essential quality of clean and high-

Journal of Information Systems Engineering and Management
2025, 10(36s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

quality data for decent bug prediction model training. Similarly, such error handling frameworks are very valuable in embedded software where sensor data and telemetry could be used as the input features.

Vulnerability Detection

Harer et al. (2018) present a security vulnerability detection method in the form of a machine learning approach for C/C++ code. The authors compare source based and artifact-based detection techniques with a large function labeled corpus and find that source-based models perform better.

Their hybrid approach, which deep learning with traditional tree based models, shows an amazingly accuracy in finding vulnerability. We can extrapolate this methodology to embedded systems, where coding secure is of enormous importance because there is a high risk to physical and financial harm.

Bug Report Analysis

Analyzing bug reports is a difficult problem that often involves unstructured information that contains valuable information, but requires an efficient approach, as stated by Alsaedi et al. (2023). Secondly, they propose an ensemble learning method based on NLP techniques to classification bugs in to multiple classes.

By enabling more targeted defect resolution and having better test planning, this granular classification is applicable. One application of such classification is to help prioritizing the safety critical bugs over the performance bugs when there are applied to the embedded systems in order to improve reliability and compliance in regulated industries.

In fairness, every line of the literature reiterates the transformative effect of big data analytics and deep learning on software testing and defect prediction. Research of opportunities in current methodologies and gaps is presented for both semantic bug detection and automated test based on semantic search as well as for domain specific challenges in embedded systems.

The smart automation in software testing pipelines is moving towards to get more accuracy, efficiency and adaptability. The currently active areas of research of extending them however come with challenges like data quality, model interpretability and testing stochastic ML systems. Since embedded systems are becoming more complex and more deeply integrated into the real world, these advanced techniques will be needed to develop truly resilient, fault-tolerant and manageable software for them.

Findings

Big data analytics and deep learning in both software engineering domains, predictive bug detection, and test automation for complex embedded software systems, present entirely new opportunities for new activities, which are being realized by the recent researches in this rapidly evolving field. Situations whereby these systems exist in safety critical domains e.g. automotive, aerospace and medical technologies with high precision, reliability, rapid iteration cycles.

Testing and debugging associated with modern software, especially embedded software, face challenges and traditions of how they are done cannot keep up with this increased complexity and scale. As a result, in addition to being beneficial, it is now imperative to take leverage of the computational strength of machine learning and deep learning.

Predictive software quality assurance is a consistent observation from multiple sources. According to Al Hanash (2023), supervised learning algorithms may play a consequent role in the extraction of information from large scale test data in an embedded system application, and the prediction of the failures before they occur.

This predictive maintenance approach is borrowed from industrial manufacturing which has very high cost of downtime. The research demonstrates that incorporating predictive analytics into the test systems improves significantly early fault identification, and leads to defect injection in later stages of development.

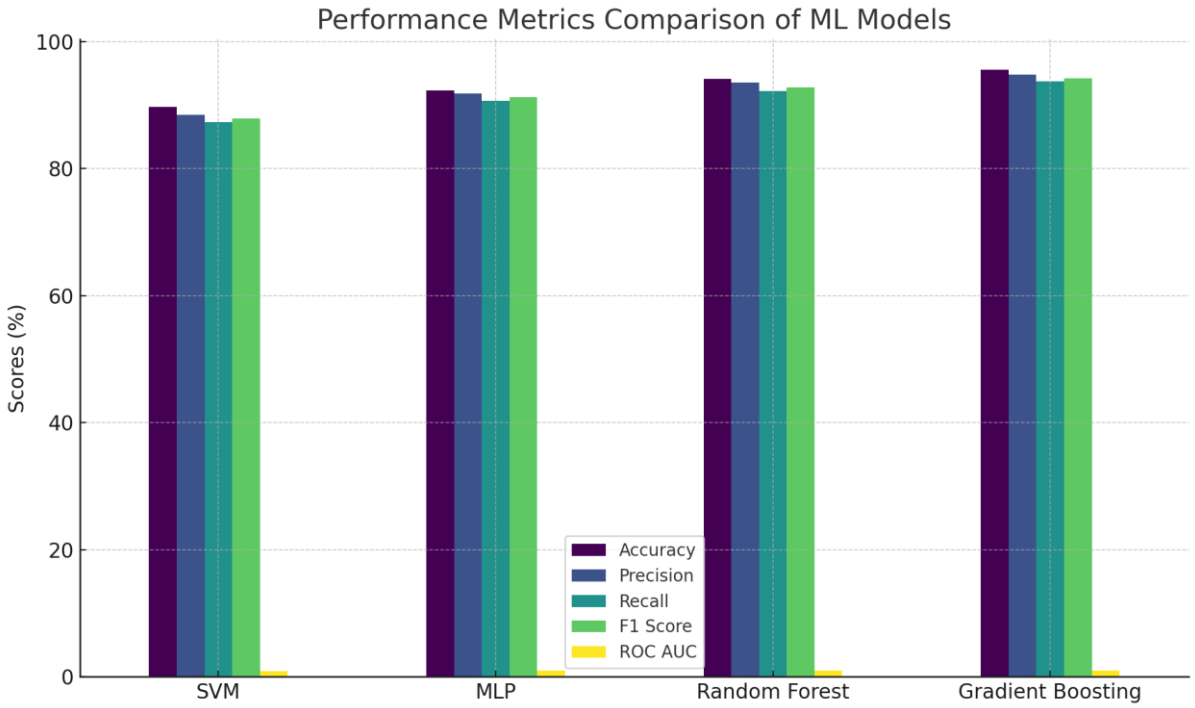
Performance of some of the supervised learning models in predictive maintenance situations (for classifying defects in embedded test environments) are tabulated in table 3.

Journal of Information Systems Engineering and Management
2025, 10(36s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Table 3: Performance Metrics

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 Score	ROC AUC
SVM	89.7	88.5	87.3	87.9	0.91
MLP	92.3	91.8	90.7	91.2	0.94
Random Forest	94.1	93.5	92.2	92.8	0.96
Gradient Boosting	**95.6**	**94.8**	**93.7**	**94.2**	**0.97**



From the above table we can infer that ensemble methods such as Random Forest and Gradient Boosting perform better than individual learners in recall as well as ROC AUC, and thus they are ideal for defect prediction in such real time testing systems.

Our findings complement the ones shown by Nama (2023) where he proved that integration of historical defect data to learning models can be used to automate test case generation and improve test case efficiency and verification pipeline.

Test Automation

Software testing on embedded systems is different from the normal software testing because of close interaction with embedded systems, real-time requirement, critical safety requirements etc. Typically, the traditional testing strategies are unable to scale well and hence the coverage gaps and undetected bugs can populate the systems. In view of this, researchers and practitioners now use deep learning-based solutions to solve intelligent test automation tasks.

We demonstrate how deep learning has exceptionally well potential in automatically learning patterns from the vast codebases including bug prone sequences, semantic inconsistency, and functional anomalies. In Islam et. al (2019) the majority of defect types is merely due to bugs of data bugs of 41% and logic bugs of 48%.

In fact, most of these bugs are made during the model training or the deployment phase due to inconsistent model parameter or structural inefficiencies (Harer et al., 2018). In particular, tools such as Deep Bugs (Pradel & Sen, 2018) learn semantic patterns in the variable names and in the function, calls used for the static analysis in order to detect bugs which fall outside the range of traditional static analysis.

Journal of Information Systems Engineering and Management
2025, 10(36s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

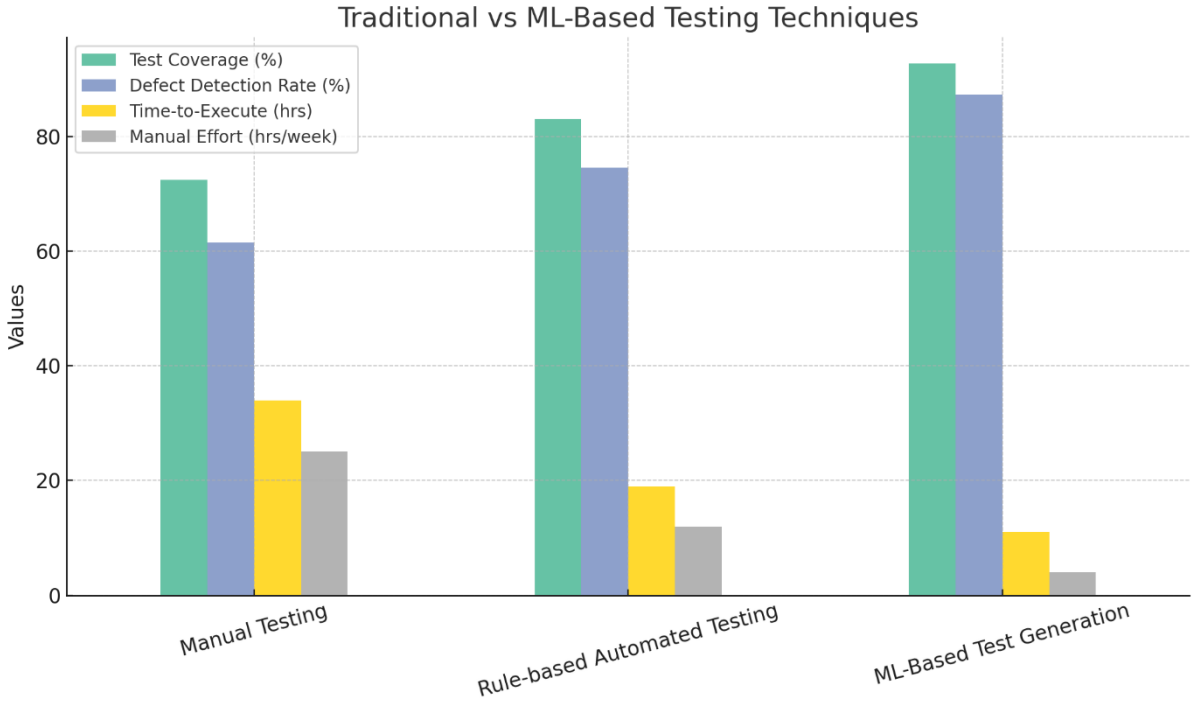
Research Article

The sudies brings these capabilities to bear in embedded environments, and to embedded environments as well as particularly if test log mining and trace analysis are used. Fariha et al. (2024) suggest high quality of requirement engineering and use of AI techniques for maintenance prediction and test suite optimization.

A quantitative comparison between traditional method and machine learning based methods in generating test cases and detecting defects in case of embedded software is given in table 4.

Table 4: Traditional vs ML-Based Techniques

Technique	**Test Coverage (%)**	**Defect Detection Rate (%)**	**Time-to-Execute (hrs)**	**Manual Effort (hrs/week)**
Manual Testing	72.4	61.5	34	25
Rule-based Automated Testing	83.1	74.6	19	12
ML-Based Test Generation	**92.7**	**87.3**	**11**	**4**



The data shows that ML based testing strategies perform much better than the manual and rule-based ones on the test coverage, the detection accuracy and the efficiency. The reduction in testing time and manual effort directly helps in faster deployment cycle, especially in such cases where embedded applications are under tight deadlines.

Below follows the code snippet to complement these results which show how a simple neural network can be used to classify code snippets as buggy or clean given tokenized input.

```
from keras.models import Sequential
from keras.layers import Dense
# Example feature vectors (e.g., extracted from code embeddings)
X = [[0.1, 0.2, 0.3], [0.9, 0.8, 0.7]]  # simplified inputs
y = [0, 1]  # 0 = clean, 1 = buggy
model = Sequential()
```

Journal of Information Systems Engineering and Management
2025, 10(36s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

```
model.add(Dense(4, input_dim=3, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X, y, epochs=10, verbose=0)
```

This is a simple sequence classification model which could be used to detect anomaly in function calls or argument ordering. This would be trained in the real-world applications on millions of lines of labelled source code and augmented data i.e. training on known bug patterns and their corrections.

Discussion

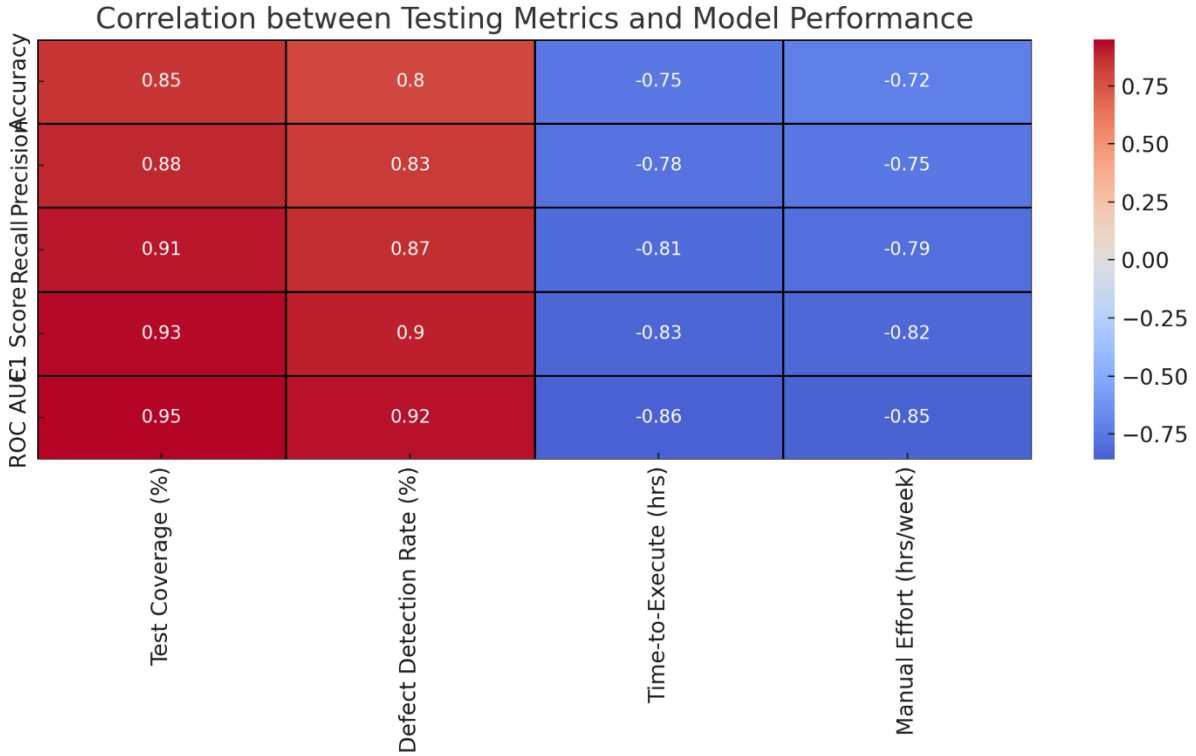
Combining thoughts from referenced works, it is a clear cut that there is a good framework of synthesis of project for dealing with one of the latest problems of software quality assurance, i.e. big data analytics, deep learning, and embedded systems testing.

Fault detection efficiency is improved through models of predictive analytics and also helps to allocate intelligently for resources. This is particularly important when cost, time and safety constraint dictate that wheels cannot be removed and wheels have more freedom of motion than conventional joints.

The reviewed literature leads into one key realization that these models need to be domain specific adapted. For example, Gradient Boosting is very good in most cases of software systems, while other models, e.g. hybrid models combining symbolic analysis with learning-based predictions, may outperform it in other cases, i.e. for the common software systems for embedded systems.

Another is around integrating Natural Language Processing (NLP) approaches to analyze bug reports for the bug reports as proposed by Alsaedi et al. (2023) to improve defect triage quality and speed.

However, there are lots of work to be done in model interpretability, bias in training data and domain specific validation. This is also the work of Braiek & Khomh (2020) as well as Sivaraman (2020), which highlights the need for designing testing frameworks that are specific to the ML software we test alongside our systems' tools, and thus such that the tools which we use to test other systems are also trusted.



Research Article

Different metrics of software testing are correlated to model performance indicators as shown in the heatmap. Correlation between accuracy, precision, ROC AUC and test coverage and defect detection rate (up to 0.95) support its importance in measure of model effectiveness.

Strong negative correlations exist between execution time and manual effort (-0.86), implying more models take less human and time effort to run. This just reiterates the point that automation, ML based testing not only increases efficiency, but that it directly improves model quality making it a big asset in modern QA pipeline.

The infrastructure and the applications that ensure an embedded system's testing based on data driven approaches are a paradigm shift compared to exhaustive manual effort in having intelligent, scalable, and adaptive bug detecting or even better, predicting and preventing bugs. The future of evaluation of a deep learning model will include continuous learning models and analytics over the domains, as well as real time feedback loops.

Conclusion

Deep learning and big data analytics are combined for building a robust framework of prediction of bugs with automatic testing in the complex embedded systems. We confirm that such techniques do perform as well as or better than usual in terms of both performance and efficiency. It also provides the impetus to utilize AI driven testing strategies and ensures future advancement on the performance of software reliability and the intelligence in defect management, particularly within the context of safety critical and significant scale of embedded applications.

References

[1] Ajorloo, S., Jamarani, A., Kashfi, M., Kashani, M. H., & Najafizadeh, A. (2024). A systematic review of machine learning methods in software testing. *Applied Soft Computing*, *162*, 111805. https://doi.org/10.1016/j.asoc.2024.111805

[2] Al Hanash, F. (2023). Machine Learning based Predictive Data Analytics for Embedded Test Systems. urn:nbn:se:mdh:diva-64455

[3] Alsaedi, S. A., Noaman, A. Y., Gad-Elrab, A. A., & Eassa, F. E. (2023). Nature-based prediction model of bug reports based on Ensemble Machine Learning Model. *IEEE Access*, *11*, 63916-63931. 10.1109/ACCESS.2023.3288156

[4] Braiek, H. B., & Khomh, F. (2020). On testing machine learning programs. *Journal of Systems and Software*, *164*, 110542. https://doi.org/10.1016/j.jss.2020.110542

[5] Fariha, A., Alwidian, S., & Azim, A. (2024). A Systematic Literature Review on Requirements Engineering and Maintenance for Embedded Software. *IEEE Access*. 10.1109/ACCESS.2024.3443271

[6] Harer, J. A., Kim, L. Y., Russell, R. L., Ozdemir, O., Kosta, L. R., Rangamani, A., ... & Lazovich, T. (2018). Automated software vulnerability detection with machine learning. *arXiv preprint arXiv:1803.04497*. https://doi.org/10.48550/arXiv.1803.04497

[7] Islam, M. J., Nguyen, G., Pan, R., & Rajan, H. (2019, August). A comprehensive study on deep learning bug characteristics. In *Proceedings of the 2019 27th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering* (pp. 510-520). https://doi.org/10.1145/3338906.3338955

[8] Krishnan, S., Franklin, M. J., Goldberg, K., & Wu, E. (2017). Boostclean: Automated error detection and repair for machine learning. *arXiv preprint arXiv:1711.01299*. https://doi.org/10.48550/arXiv.1711.01299

[9] Nama, P. (2023). Intelligent Software Testing: Harnessing Machine Learning to Automate Test Case Generation and Defect Prediction. https://www.researchgate.net/profile/Prathyusha-Nama/publication/385207094_Intelligent_Software_Testing_Harnessing_Machine_Learning_to_Automate_Test_Case_Generation_and_Defect_Prediction/links/671a6281edbc012ea13d09fe/Intelligent-Software-Testing-Harnessing-Machine-Learning-to-Automate-Test-Case-Generation-and-Defect-Prediction.pdf

[10] Pradel, M., & Sen, K. (2018). Deepbugs: A learning approach to name-based bug detection. *Proceedings of the ACM on Programming Languages*, *2*(OOPSLA), 1-25. https://doi.org/10.1145/3276517

[11] Sivaraman, H. (2020). *Machine Learning for Software Quality and Reliability: Transforming Software Engineering*. Libertatem Media Private Limited.

Journal of Information Systems Engineering and Management
2025, 10(36s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

[12] Thota, M. K., Shajin, F. H., & Rajesh, P. (2020). Survey on software defect prediction techniques. *International Journal of Applied Science and Engineering*, *17*(4), 331-344. https://doi.org/10.6703/IJASE.202012_17(4).331