Artificial Intelligence For Software Testing-Perspectives And Practices

Nisha Jha
Research Scholar
Department of Computer Engineering
J.C. Bose University of Science and Technology, YMCA
Faridabad
nishajha1992@gmail.com

Rashmi Popli
Assistant Professor
Department of Computer Engineering
J.C. Bose University of Science and Technology, YMCA
Faridabad
rashmimukhija@gmail.com

Abstract—**Artificial Intelligence (AI) has emerged as a buzzword for current software applications. The modern advancements in the Information Technology sector have invigorated the need to incorporate AI competencies into software services. This objective has constrained the organizations to revisit their software development processes. Software testing plays a vital role in validating the software quality. Both the AI and software testing researchers and Practitioners must lead the innovations to address their integrating challenges. AI and Machine Learning (ML) have the potential to advance the capabilities to test the software intensely. The objective of the paper is to review the state-of-the-art of applying AI in software testing briefly. It also discusses the software testing activities mapped to AI and the related challenges.**

Keywords—Software Testing; Machine Learning; Artificial Intelligence; Test Automation

I. Introduction

Current automated software testing techniques are beneficial to enhance testing efficacy at a reasonable cost; a lot of manual work is still needed. To accomplish high software quality, both manual, as well as automated testing approaches are required. To cope with the latest software advancements, test objects must be consistently updated to stay pertinent [5]. Current test automation can't sum up across all applications and can't impersonate human knowledge. A substantial gap exists between the present state of practice and a fully automated approach to software testing.

Software testing is primarily an activity in applying sample inputs to a system and estimating the yields to decide the accuracy of the application's conduct. This testing input and output function is quite similar to the elementary functionalities of training and executing modern AI and ML systems [4]. The similarity suggests that the field of Software Testing is ready to profit from ongoing advances in AI and ML.

According to Wikipedia [21], AI research can be defined as "the study of intelligent devices that learn from its environment and act accordingly in order to maximize the chance of successful accomplishment of its goals." AI makes a machine learn from the data and think about the decisions precisely.

ML techniques, which have been in existence for a long time, vested the software systems to furnish extra beneficial features with the extensive accessibility of digitized information and computations [15]. Autonomous driving, natural language processing, or image recognition are exquisite applications of ML that can be realized in numerous sectors, including social networking, finance, healthcare, manufacturing, etc.

Conventionally, software systems are developed using a deductive approach, in which rules are written to govern the system behaviors as program code [6]. Whereas in ML systems, the rules are inferred from training data using an inductive approach. The shift in the paradigm about ML-based software applications' conduct has resulted in software systems that are intrinsically difficult to be tested and verified.

AI-based software testing describes the power and applications of AI algorithms and solutions for automatically streamlining the process of software testing in fault detection and analysis, test case generation, selection and execution, and quality analysis [11]. It comprises diverse testing activities in AI-based software testing.

There is a necessity for a holistic perspective of designing software-intensive frameworks with ML competencies in a realistic environment. Many researchers and industry practitioners from software engineering (SE) and ML have emphasized the necessity for such a holistic view.

A. Software Testing

Software testing has always been a critical phase of SE [16]. Automated software testing can help in accomplishing effective testing with reasonable costs.

Test cases are one of the essential elements of the software testing activity. A test case represents the conditions under which the system under test is executed in order to find a bug. When a test case uncovers a failure, the test is considered as successful. A test case takes the variable input values for execution. To assess the results of test cases, expected outcomes and actual outcomes are evaluated and compared. The element used to verify the accuracy of the outputs produced is termed as test oracle [13].

In the past decades, substantial research has been performed to automatize test case generation, test selection, test oracles, etc. Rapid growth has been observed in practices emphasizing the usage of automated software testing tools.

B. Machine Learning

There are mainly three classes of ML: supervised, unsupervised, and reinforcement learning (Fig. 1.).
A labeled training dataset is used to build an ML model in supervised learning, such as classification and regression problems.
Unsupervised learning like clustering and association rule learning, the model uses unlabeled data to discover new patterns.
In reinforcement learning, an agent learns from the environment and makes decisions. One of the compelling examples of reinforcement learning is video games.

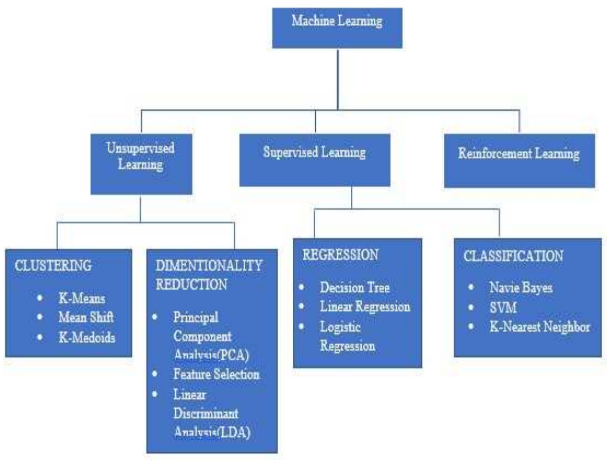


Fig. 1. Machine Learning Techniques

Reinforcement Learning aims to reach a goal by learning from experience even in the absence of training datasets. Q-Learning and Temporal Difference are some of the examples of commonly used algorithms [18].
A wide variety of AI techniques can be applied to manage software testing activities. The most widely used techniques are Artificial Neural Networks, Support Vector Machines, k-Nearest Neighbor, Naive Bayes Classifier, and Decision Trees.

II. Literature Review

Machine Learning can be treated as one of the best test approaches to provide exponential test coverage [4]. It includes the following steps:
1. ML models can be trained accordingly to generate test input and test validations for various applications.
2. ML provides the feature of executing the same test cases for different applications.
3. ML test generation and execution can automatically improve storage, network, and compute in the cloud.

Machine learning has for quite some time been utilized for different purposes in software testing. Briand [3] gave a concise outline of the cutting edge and reports on various novel applications in the area of software testing on the basis of personal experience.

ML testing is used to allude to any action pointed toward distinguishing contrasts among existing and required practices of AI frameworks. ML testing is not the same as testing approaches that use ML or those guided by machine learning, which should be referred to as 'machine learning-based testing'[14].

Some ML applications are expected to learn properties of data sets where the accurate answers are not definitely known to human clients. It is quite challenging to test such ML systems due to the lack of reliable test oracles. Murphy et al. [2] portray a software testing approach pointed toward tending to this issue by testing implementations of two various ML ranking algorithms: Support Vector Machines and MartiRank.

A design science approach had been employed [7] to examine how machine insight can be utilized to improve the automation of the analysis of non-functional testing. A model was created to demonstrate the ability of machine intelligence methods to provide helpful information about the relationships between different test cases and their histories.

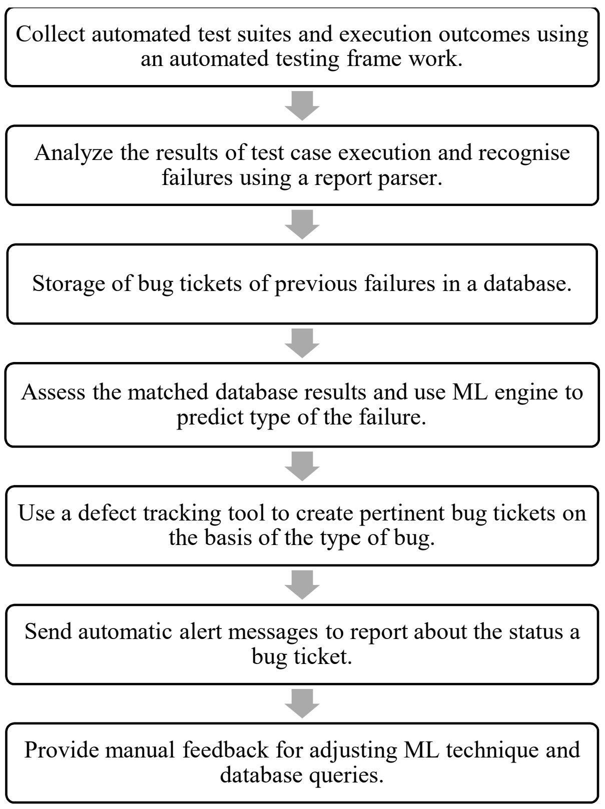
Many teams at Microsoft have invested massive energy into fostering an extensive portfolio of AI applications and stages by incorporating AI into existing software engineering measures and developing ML ability. Amershi et al. [10] depicted the consequences of an examination to get familiar with the cycle and practice changes attempted by various Microsoft groups as of late.

''Testing AI software deliberates the several testing tasks for AI-based frameworks. Precise and feasible quality validation models, approaches and procedures should be created and applied for AI-based software to support the testing activities to accomplish pre-defined test requirements and meet adequate testing criteria in order to comply with quality assurance standards.'' Therefore, testing AI features of the software comprises diverse testing activities to discover software bugs and validate the software's performance. Testing aims to realize well-defined test prerequisites, meet pre-defined testing criteria and guidelines of quality assurance of the AI software under-test [11]. The primary focal points of AI software testing are summed up as follows:
(a) Testing AI functional highlights to guarantee satisfactory quality, accuracy, consistency and practicality using AI techniques.
(b) Testing AI software's quality on the basis of well-defined quality guidelines and evaluation models like reliability, scalability, security, performance and availability.
(c) Applying data-driven AI methods to ease automated software testing and AI testing procedures.

III. Artificial Intelligence and Software Testing

This section comprises a journey of leveraging prevailing AI research and its mapping to relevant software testing activities and challenges.

Fig. 2. demonstrates an approach for ML-based test automation according to one or more embodiment of the present invention [20].



Collect automated test suites and execution outcomes using an automated testing frame work.

Analyze the results of test case execution and recognise failures using a report parser.

Storage of bug tickets of previous failures in a database.

Assess the matched database results and use ML engine to predict type of the failure.

Use a defect tracking tool to create pertinent bug tickets on the basis of the type of bug.

Send automatic alert messages to report about the status a bug ticket.

Provide manual feedback for adjusting ML technique and database queries.

Fig. 2. Automated software testing using ML approach

A. Software Testing Activities Supported by AI

Different types of activities can be mapped to AI and ML algorithms to smoothen the process.

Fig. 3. summarizes the various software testing activities supported by AI.

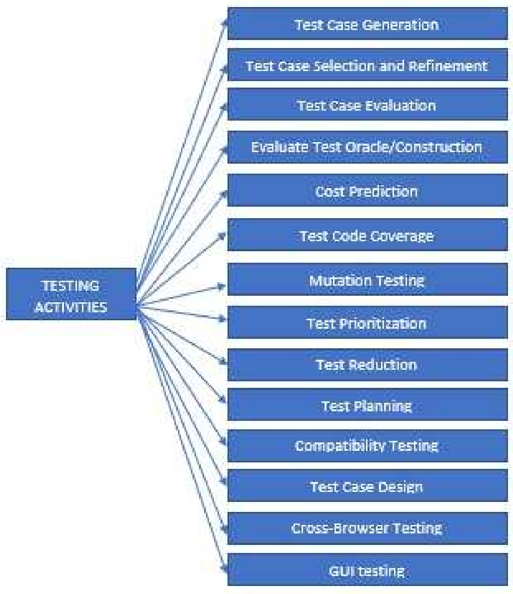


Fig. 3. Current AI Software Testing Coverage

B. Benefits

Existing methodologies are not able to provide complete solutions for 100% automation in software testing, and a significant amount of manual testing is still essential. There is a considerable gap between the performance of testing done by humans and machines. Human analyzers can check an application's condition, act wisely, and investigate uncovered bugs. In order to advance effectiveness and diminish the expenses, there is a requirement to improve automated software testing procedures and present more wise computerized testing conduct that is equipped for mirroring human actions [5].

Zhang [1] elaborates some of the advantages of applying ML for testing activities validation and verification, test oracle generation, test adequacy matrix, defect and cost prediction.

Applying the progressions in AI and ML will help testing find improvement propels, increment the effectiveness, speed and adequacy of programming testing, and free analyzers from numerous routine testing errands [8]. When used to extract the test input, execution, and assessment issues, AI and ML empower testing at scale, normalization of quality measurements, benchmarking, and a worldwide arrangement of reusable experiments.

During the implementation and testing phase, AI provides automatic troubleshooting or debugging and error tracing procedures. It can also support the inclusion of distinct software procedures into extensive architectures [17].

Ferreira et al. [19] focus on gaining the momentum of ML for software testing tasks. ML gives the best results with unstructured and unlabeled data without Feature Engineering; ML provides high-quality results with sequence prediction.

The AI predictive analytics can be used as a crucial technique to realize all the possible test cases and create the software applications better in terms of quality, robustness, reliability and performance [12].

The main benefits have been summarized in Fig. 4.

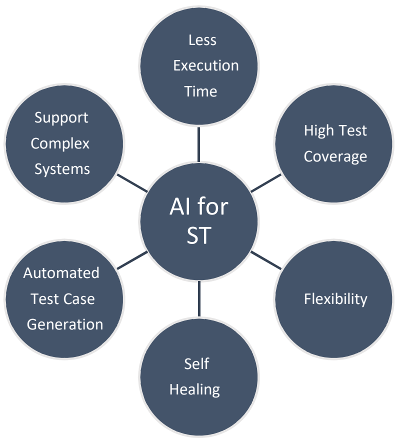


Fig. 4. Benefits of Applying AI to Software Testing

C. Challenges

Developing ML systems in a realistic environment is quite difficult due to the high complexities in engineering conventional systems.
The significant challenges of applying AI and ML in ST are:

- Domain Knowledge Gap Problem [5][8]
- Training Data Availability [6]
- Oracle Problem [9]
- Full Automation
- Scalability of Testing
- Test Management
- 100% Accuracy
- Quality Assurance
- Computational cost, Size and quality of the dataset [19]
- Test Case Design [6]
- Test Result Interpretation

Unfortunately, a rift has been observed between these two respective communities. One of the reasons is AI community focus on algorithms and the related performance characteristics [6]. Whereas stakeholders in the SE community center around the implementation and deployment of these algorithms or techniques.

IV. Applying AI To Software Testing

The section demonstrates the application of AI for software testing through examples. The First AI research area has been focused on, and then the research has been mapped to testing.

The example uses text generation to aid in software testing. Text generation is one of the leading applications of natural language processing. The aim is to automatically generate natural language texts using used knowledge in computational linguistics and AI.

Being able to perceive objects, the next important step is to make the machine learn the test flows from human testers. A test can be defined as a series of actions to be performed on a system under test and a set of anticipated outcomes.

Each step in the test flow execution process has been categorized into three categories at a broader level, i.e., Perceive, Act, Observe. Perceive class focuses on creating pre-defined situations for a test flow, concluding the environment, and planning actions accordingly. The task of the Act category belongs to the establishment and execution of appropriate measures on the basis of the former created plan. Efforts belonging to the Observe class enter around comparing expected results to the system's actual behavior under test and checking the precision. The steps are repeated continuously in the testing procedure by letting the Observe class start a new Perceive phase every time, forming a cycle. The presentation of the test flow using this approach is the initial move to address the machine language. The next step is to outline a language to communicate substantial test flows. The language should be adequately expressive to cover essential test cases in the test flow generation phase. The language should also be able to make utilization of webpage component abstractions of the system.
Fig. 5. below shows a sample form and related test flows.

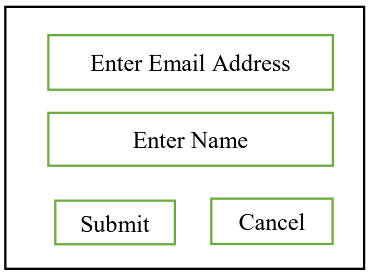


Fig. 5. Sample Form

Test Flows:
Observe: Signup Form
Focus Textbox-Enter Email Address
Try valid entry
Observe: Success Message

Observe: Signup Form

Focus Textbox-Enter Email Address
Try invalid entry
Observe: Error Message

The components, actions, and observations are the primary building blocks of the language. Components comprise the web page elements. Observations are used to represent information about the web page components. The actions are the measures performed on the components. By interleaving observations and actions, the language allows for the test flows specification. The language's dynamic learning objects are utilized instead of specific inputs that may be only relevant to a solitary system under test in a particular domain application. Fig. 6. below shows a shopping cart scenario to create test flows.

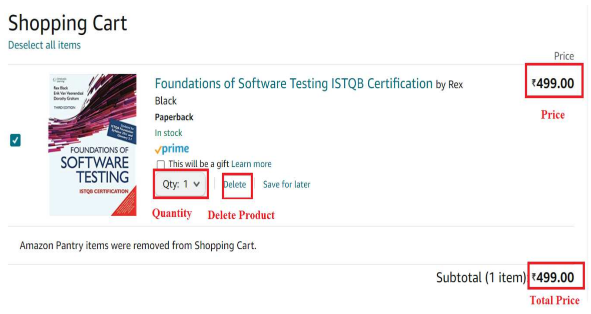

Shopping Cart
Deselect all items
Price
Foundations of Software Testing ISTQB Certification by Rex Black
Paperback
In stock
prime
This will be a gift Learn more
Qty: 1
Delete
Save for later
₹499.00
Price
Quantity
Delete Product
Amazon Pantry items were removed from Shopping Cart.
Subtotal (1 item) ₹499.00
Total Price

Fig. 6. Sample of Shopping Cart

Test Flows for the above scenario are:
Observe: Shopping Cart
Focus Added Item
Try Click Delete
Observe: Item not in the cart

Observe: Shopping Cart
Focus Added Item
Try Change in Quantity
Observe: Total Price Change

The goal is to share the experiences and later realize them as a framework for future work.

V. Conclusion

The software will continue to be a part of human life as far as the future can be envisioned. AI and ML will drastically revolutionize the idea of software testing and software quality in the upcoming years, far more radically than most expect. Researchers and industry professionals from the two domains have to realize the opposite side's concerns and have an all-encompassing perspective on designing ML frameworks. Software Testing and ML communities should cooperate to resolve the critical challenges to ensure the quality of AI-based software systems in general and benefit from each other.

This paper can aid as a base to acquire a holistic view and a treasury of papers, analysts, and associations to investigate this subject.

References

[1] D. Zhang and J. J. P. Tsai, "Machine learning and software engineering," in *Proceedings of the International Conference on Tools with Artificial Intelligence*, 2003, no. May, pp. 22–29, doi: 10.1109/tai.2002.1180784.

[2] C. Murphy, G. Kaiser, and M. Arias, "An approach to software testing of machine learning applications," in *19th International Conference on Software Engineering and Knowledge Engineering, SEKE 2007*, 2007, pp. 167–172.

[3] L. C. Briand, "Novel applications of machine learning in software testing," in *Proceedings - International Conference on Quality Software*, 2008, pp. 3–10, doi: 10.1109/QSIC.2008.29.

[4] J. Arbon, "AI for Software Testing," *PNSQC Proc.*, pp. 1–19, 2017.

[5] D. Santiago, T. M. King, and P. J. Clarke, "AI-Driven Test Generation: Machines Learning from Human Testers," in *In Pacific Northwest Software Quality Conference (ed.). Proceedings of the 36th Pacific NW Software Quality Conference. Portland: Pacific Northwest Software Quality Conference. https://www.pnsqc.org/wp-content/uploads/2018/09/38-Santiago-AI-Driven-Test-Ge*, 2018, pp. 1–14, [Online]. Available: https://www.pnsqc.org/wp-content/uploads/2018/09/38-Santiago-AI-Driven-Test-Generation.pdf.

[6] F. Khomh, B. Adams, J. Cheng, M. Fokaefs, and G. Antoniol, "Software Engineering for Machine-Learning Applications: The Road Ahead," *IEEE Softw.*, vol. 35, no. 5, pp. 81–84, 2018, doi: 10.1109/MS.2018.3571224.

[7] E. Wallengren and R. S. Sigurdson, "Machine Intelligence in Automated Performance Test Analysis," 2018.

[8] A. Bennaceur and K. Meinke, "Machine Learning for Software Engineering -Models, Methods, and Applications," in *2018 ACM/IEEE 40th International Conference on Software Engineering: Companion Proceedings*, 2018, vol. 11026 LNCS, no. 1, pp. 3–49, doi: 10.1007/978-3-319-96562-8_1.

[9] S. Sherin, M. U. Khan, and M. Z. Iqbal, "A systematic mapping study on testing of machine learning programs," *arXiv*, 2019.

[10] st. al Saleema Amershi, Andrew Begel, "Software Engineering for Machine Learning: A Case Study," in *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP '19). IEEE Press*, 2019, no. May 2019, pp. 291–300, [Online]. Available: https://fontysblogt.nl/software-engineering-for-machine-learning-applications/.

[11] C. Tao, J. Gao, and T. Wang, "Testing and Quality Validation for AI Software-Perspectives, Issues, and Practices," *IEEE Access*, vol. 7, pp. 120164–120175, 2019, doi: 10.1109/ACCESS.2019.2937107.

[12] H. Hourani and A. Hammad, "The Impact of Artificial Intelligence on Software Testing," in *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology, JEEIT 2019 - Proceedings*, 2019, pp. 565–570.

[13] V. H. S. Durelli *et al.*, "Machine learning applied to software testing: A systematic mapping study," *IEEE Trans. Reliab.*, vol. 68, no. 3, pp. 1189–1212, 2019, doi: 10.1109/TR.2019.2892517.

[14] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine learning testing: Survey, landscapes and horizons," *arXiv*, pp. 1–37, 2019, doi: 10.1109/tse.2019.2962027.

[15] G. Giray, "A software engineering perspective on engineering machine learning systems: State of the art and challenges," *arXiv*, no. 1, 2020.

[16] J. J. Li, A. Ulrich, X. Bai, and A. Bertolino, "Advances in test automation for software with special focus on artificial intelligence and machine learning," *Softw. Qual. J.*, vol. 28, no. 1, pp. 245–248, 2020, doi: 10.1007/s11219-019-09472-3.

[17] M. Barenkamp, J. Rebstadt, and O. Thomas, "Applications of AI in classical software engineering," *AI Perspect.*, vol. 2, no. 1, pp. 1–15, 2020, doi: 10.1186/s42467-020-00005-4.

[18] R. Lima, A. M. R. Da Cruz, and J. Ribeiro, "Artificial Intelligence Applied to Software Testing: A Literature Review," *Iber. Conf. Inf. Syst.*

Technol. Cist., vol. 2020-June, no. June, 2020, doi: 10.23919/CISTI49556.2020.9141124.

[19] F. Ferreira, L. L. Silva, and M. T. Valente, “Software Engineering Meets Deep Learning: A Mapping Study,” *arXiv*, 2020, [Online]. Available: http://arxiv.org/abs/1909.11436.

[20] S. Mahapatra and S. Mishra, “Usage of Machine Learning in Software Testing,” no. August, pp. 39–54, 2020, doi: 10.1007/978-3-030-38006-9_3.

[21] https://en.wikipedia.org/wiki/Artificial_intelligence