Software Testing Using Artificial Intelligence: A State of Art

[1] Suman Motton, [2] Parminder Kaur
[1]Research Scholar, [2]Associate Professor
[1]Department of Computer Science, Guru Nanak Dev University, Amritsar,
[2]Department of Computer Science, Guru Nanak Dev University, Amritsar,
[1] sumancs.rsh@gndu.ac.in, [2] parminder.dcse@gndu.ac.in

ABSTRACT

Artificial Intelligence (AI) emerges as the latest technology across all software industries as well as domains. It is being leveraged in the field of software testing to ease the automation testing process and deliver more quality outcomes. The application of AI in software testing will make the entire testing process faster, clearer, easier, and within budget. This paper makes an effort to elaborate on the significance of performance testing in the field of software testing using AI. This paper represents a comprehensive review of AI/ML techniques in software testing.
Keywords: software testing, artificial intelligence, machine learning, black-box testing, performance testing

INTRODUCTION

The software has entered practically every industry sector and has grown increasingly. If the software satisfies all of the user's needs, then it is considered to be high-quality software. However, it is impossible to build high-quality software without testing. Significant losses result from poor-quality software. The system failure industry may be highly expensive, and important systems (flight control, nuclear reactor monitoring, medical applications, and so on) can result in human lives being lost. Software testing is the process of executing an issue to find errors in the code. It is the process of assessing a system component manually or automatically to ensure that it fulfills specified criteria. Testing is essential to identify differences between predicted and actual outcomes. Two elements reflect the excellent quality of the product. The first is verification, and the second is validation. During the development phase, verification and validation were carried out. Software testing is the portion of validation and verification that involves evaluating and analyzing program code. It is one of the two most expensive stages of the software development lifecycle. Barenkamp et.al [1] mentioned SDLC's objective is to complete software in the development phase. It is difficult to produce excellent quality software without the testing process. Good quality items should be devoid of flaws. Only the testing process can uncover software problems and defects. That is the main reason for selecting the software testing phase of the SDLC life cycle model above the other phases. There are several approaches to software testing, however, the most common methods rely on common processes. During the implementation phase, modules are informally checked while being coded by the programmer; this is known as desk checking.

TYPES OF SOFTWARE TESTING

After the programmer is confident that the module looks correct, a separate testing team does thorough testing on the module. There are two basic types of methodical testing:

A. Non-Execution based

Non-execution-based testing relies on a fault-detection strategy. The fault-detecting power of these non-execution-based techniques leads to rapid thorough, and early fault detection.

B. Execution-based testing

The modules get executed against test cases in this type of testing.

- Black-Box testing: The code itself is ignored. The only information used in designing test cases is the specification document.
- White-Box testing: The code itself is tested without regard to the specifications.

BLACK-BOX TESTING TYPES

A. Functional Testing: It defines the software functional requirements of the system.

B. Regression Testing: It ensures that the new code is backward compatible with the old code. In other words, a new software update does not affect how the software works. This is accomplished by doing system maintenance and updates.

C. Non- Functional Testing (NFT): It is used to check non-functional aspects (performance, usability, reliability) of software applications. Non-functional testing is as important as functional testing. Non-functional testing parameters are performance testing, availability, reliability testing, survivability, flexibility, reusability, interoperability testing, and portability. This research work further focuses on performance testing.

PERFORMANCE TESTING

Performance testing is a critical component of software development. The primary purpose of performance testing is to establish how the software behaves during peak hours or under strong workloads. Performance testing entails more than just assessing the software or application. It is also about determining whether or not standards and benchmarks have been met. Accuracy is regularly checked, evaluated, and tested throughout performance testing. As a result, the objective is to not only uncover system problems but also to avoid future bottlenecks.

- **Why there is a need for performance testing?**

The performance and responsiveness of an application are crucial in today's industry. Organizations may use performance testing to determine the efficiency and efficacy of a

product's targeted goal in real-time. This test helps to reduce production costs by increasing knowledge of any defects in the product or factors that may create problems in the future. The danger of a system going live without adequate performance testing is that it will experience issues such as sluggish performance when more people utilize it. This affects usability and sales objectives.

SEARCH PROCESS

Although ML techniques have been present for more than 40 years, interest in using these algorithms to address real-world issues outside of the field of AI has increased significantly during the last three decades.

The primary search method was automatic searching. To locate as much significant primary research as possible. As looked at the four digital libraries that collectively include the majority of the software testing literature. Specifically searched SpringerLink, IEEE Digital Library, ScienceDirect, and ACM Digital Library. Then included Web of Science as well to avoid the necessity to search numerous publisher-specific sites. ACM, Elsevier, IEEE, Springer, and Wiley are just a few of the digital libraries whose articles are indexed by the generic indexing service Web of Science. Mainly played around with various search string combinations by connecting them using Boolean operators. (i.e., AND and OR). The given work employs a two-part search string: the first portion contains all keywords associated with software testing, while the second half is made up of ML-related phrases. Using the Boolean operator AND, the two components are connected. The following keyword combo was thought to be the best fit for this paper.

INCLUSION AND EXCLUSION CRITERIA

Retrieved publications were sorted by the inclusion and exclusion criteria listed below.
The inclusion criteria were:
IC1: publications written in English
IC2: only primary studies
IC3: publications from the software testing domain
IC4: publications that describe the application of artificial intelligence techniques
The following exclusion criteria were specified:
EC1: publication types such as a book, book chapter, encyclopedia.
EC2: papers issued before publication of (2008)

The previously stated standards were carefully applied throughout numerous stages. Duplicate and incomplete articles were disregarded at the initial round of the review. Additionally, every publication was sorted according to year, language, and publishing type. The number of discovered papers drastically decreased from 123 to 89 after duplicates were eliminated and the criteria (IC1, EC1, EC2) were applied. The following stage entailed sorting papers based on what was written in their abstracts and titles. The result was a reduction in the number of papers included to 66. Reading the papers in their entirety was the next step in the review process. Only

the most recent or, the most detailed publication was used in the final set. The selection of 21 papers was then narrowed down using the inclusion and exclusion criteria mentioned above.

RELATED WORK

The process of reviewing software to detect errors is referred to as software testing. Software testing may also be used to assess software quality factors such as dependability, usability, integrity, security, capability, efficiency, portability, maintainability, compatibility, and so on. Software testing is a time-consuming and demanding task. AI approaches are being investigated by numerous research efforts to tackle the aforementioned challenges. Such strategies are used to improve the software's efficiency and effectiveness.

Barenkamp et.al [1] have presented the role of AI techniques in various phases of software development (Project planning, problem analysis, software design, software implementation, software testing, and integration, and software maintenance, etc.) Then, Job [2], Hammad et.al [3], Sikka et.al [4], Shanmugam et.al [5], Durelli et.al [6], Jayakumar et.al [7] have also discussed the role of AI techniques and tools in software testing. Ahmed et.al [8] have compared automated tools for software testing and reported that Selenium and SAHI performed well amongst the other tools. Ahmed [9] has mentioned various problems faced by traditional techniques and reported the solution by implementing AI techniques in software testing. Sugali et.al [10] have explained ten challenges faced by software testers while implementing AI/ML techniques. Sundaram [11] has explained the latest software testing trends like agile methodology, cloud computing, mobile applications, automation, DevOps, Big data, etc. The author has discussed challenges associated with trends such as financial risks, security, and scalability. *There is a need for more standard generalized software testing models to overcome the challenges faced in testing.* Marijan et.al [12], and Farooq et.al [13] have explained some of the biggest challenges faced by software testers while applying AI to testing. Larkman et.al [14] have proposed fuzzy cognitive Maps (FCMs) to evaluate the decision support framework for testing software. Srivastava et.al [15] have employed genetic algorithms for improving software testing efficiency. The authors have compared the genetic algorithm with exhaustive and local search and concluded that the genetic algorithm performs better than existing techniques. Vergilio et.al [16] have presented a machine learning-based approach to cluster test data with similar functionality. Sharma et.al [17] have proposed a T-Model to cover 4S Quality metrics based on an empirical study of the root cause of software failure. A lack of testing of non-functional parameters is the root cause of software failure. Jung et.al [18] have analyzed automating root causes via machine learning in agile software testing environments. It was proved that the ANN method has achieved 88.9% accuracy. Prototype or Baseline approach for extracting expert knowledge and adapting it to machine learning techniques for root cause analysis in an agile environment. Shuo et.al [19] have presented an automatic test oracle generation method using the ANN model to design a test oracles generator model. Bansal et.al [20] have resolved test oracles problems using neural networks and decision tree models. Wu

et.al [21] have constructed a test oracle for reactive systems without explicit specification using machine learning techniques.

S. No.	**Paper Reference**	**Techniques Used**	**Strengths**	**Limitations**	**Future Work**
1.	[2]	AI/ML	Quality enhanced and reducing time and cost	User interface (UI) testing is the most expensive testing	Need to Enhance practical aspects
2.	[3]	Machine learning and neural language processing (NLP)	Reduce time and increase the efficiency	Need more studies to investigate other testing areas	Use technology like Cloud technology, IoT, Big data
3.	[4]	AI (Ant colony optimization, genetic algorithm, and Bee colony)	Provide reliable and quality-assured software product	Researchers can't sure about the quality of the product until measuring quality	Meta-heuristic approaches need to be used to achieve quality
4.	[5]	Replication with validation (RV) and Safe Adaptation with validation (SVA)	RV can be distributed to a separate computational node	Two AI-based testing challenges are non-determinism and fuzzy oracles	Need more studies in the area of AIST (Artificial intelligence for Software testing)

5.	[6]	AI/ML	Scale very well thus it can be used to support increasingly complex testing activities	ML algorithms might blur the roles of testers and data scientists.	Need a collaboration between researchers and practitioners
6.	[7]	AI/ML	Testing can be made more efficient and smarter with the help of AI	Faces the challenges and finds solutions to those challenges	AI can be used for mobile application testing
7.	[8]	Software under test (SUT)	Increase efficiency and effectiveness	Challenges for designing and developing automated testing tools	Need more automated tools that will increase the system's capacity
8.	[9]	AI (Ant colony optimization, genetic algorithm, and Bee colony)	Eliminating human error and producing more reliable results	Low test coverage and high execution time	AI techniques need to take it to full potential so that it will meet the future needs of software quality
9.	[10]	AI/ML	Reduce cost and Increase Performance speed	Data privacy and overfitting	Seeking solutions that overcome the problems
10.	[11]	AI/ML	Focused on the quality of a software	financial risks, security, and scalability	Need more generalized standards for software testing models
11.	[12]	ML	Ensure the correctness and	Fuzzing not supporting	Need automated test oracles and

			trustworthiness of ML-enabled systems	multi-Criteria test generation	coverage metrics for ML model
12.	[13]	AI/ML	Check bugs and errors in a much faster way	Exhaustic search space leads to generality loss	Expansion to be performed on large data sets
13.	[14]	Fuzzy cognitive Maps (FCMs)	Provide a better understanding of AI techniques	Unable to capture all concepts from the software testing domain	Need to apply FCMs to increasingly granular representations of the AI testing framework
14.	[15]	Genetic algorithm	Improving software testing efficiency	An infinite number of paths due to loops	Need to Compare GA-selected paths into large test data
15.	[16]	Machine learning	Automatic determination of equivalence classes	Information resulting from different testing techniques is not integrated	Future work is to Generate test data
16.	[17]	AI	Focused on security scanning and Performance testing	-	More focus needs to be done on security and performance testing at the architecture level, cloud, data, and application layers
17.	[18]	ML	Given satisfactory results	Five errors were obtained functional,	Future research directions aimed to make the machine's

				connectivity, infrastructure, software test code, and interrupted test.	output less dependent on an a posteriori human validation
18.	[19]	ML	ANN can fulfill the test oracles automatic generation work that done with formal specification techniques	For some problems could not apply this method directly.	Need more focus on analyzing the model's applicable conditions, and making it more convenient to use.
19.	[20]	AI	provides maximum accuracy	The decision tree model can be used only for small programs	A decision tree model will be used for complex data types
20.	[21]	ML	Investigate issues in machine learning techniques	-	More research is needed for the testing of general reactive systems.

Table (1) - Related Work On AI/ML In Software Testing

CONCLUSION

The paper presents a comprehensive review of AI/ML techniques in software testing. Although, existing machine learning techniques have achieved satisfactory performance in testing. But still, there is a need to explore more techniques. Software testing has the huge limitation that it is a very expensive process. That is why such AI techniques should be applied so that the expenditure can be reduced. During the process of developing software, there is a shortage of

information in the literature about methods for assessing the performance of mobile applications. In the future direction, this research work further needs to explore the performance of automated techniques in mobile applications.

REFERENCES

[1] 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.

[2] M. A. Job, "Automating and Optimizing Software Testing using Artificial Intelligence Techniques," vol. 12, no. 5, pp. 594–603, 2021.

[3] H. Hourani, A. Hammad, and M. Lafi, "The impact of artificial intelligence on software testing," *2019 IEEE Jordan Int. Jt. Conf. Electr. Eng. Inf. Technol. JEEIT 2019 - Proc.*, pp. 565–570, 2019, doi: 10.1109/JEEIT.2019.8717439.

[4] N. Bhateja and S. Sikka, "Achieving quality in automation of software testing using ai based techniques," *Int. J. Comput. Sci. Mob. Comput.*, vol. 6, no. 5, pp. 50–54, 2017, [Online]. Available: www.ijcsmc.com

[5] T. M. King, J. Arbon, D. Santiago, D. Adamo, W. Chin, and R. Shanmugam, "AI for testing today and tomorrow: industry perspectives," *Proc. - 2019 IEEE Int. Conf. Artif. Intell. Testing, AITest 2019*, pp. 81–88, 2019, doi: 10.1109/AITest.2019.000-3.

[6] V. H. S. Durelli, R. S. Durelli, S. S. Borges, A. T. Endo, and M. M. Eler, "Machine Learning Applied to Software Testing : A Systematic Mapping Study," pp. 1–24, 2019, doi: 10.1109/TR.2019.2892517.

[7] N. Mulla and N. Jayakumar, "Role of Machine Learning & Artificial Intelligence Techniques in Software Testing," vol. 12, no. 6, pp. 2913–2921, 2021.

[8] S. K. Alferidah and S. Ahmed, "Automated Software Testing Tools," *2020 Int. Conf. Comput. Inf. Technol. ICCIT 2020*, pp. 183–186, 2020, doi: 10.1109/ICCIT-144147971.2020.9213735.

[9] N. Ahmed, "Old Testing Automation techniques are lagging : Artificial Intelligence has the pace," pp. 1–5, 2017.

[10] K. Sugali, C. Sprunger, and V. N. Inukollu, "Software Testing: Issues and Challenges of Artificial Intelligence & Machine Learning," *Int. J. Artif. Intell. Appl.*, vol. 12, no. 1, pp. 101–112, 2021, doi: 10.5121/ijaia.2021.12107.

[11] A. Sundaram, "Technology Based Overview on Software Testing Trends, Techniques, and Challenges," *Int. J. Eng. Appl. Sci. Technol.*, vol. 6, no. 1, pp. 0–5, 2021, doi: 10.33564/ijeast.2021.v06i01.011.

[12] D. Marijan and A. Gotlieb, "Software testing for machine learning," *AAAI 2020 - 34th AAAI Conf. Artif. Intell.*, pp. 13576–13582, 2020, doi: 10.1609/aaai.v34i09.7084.

[13] Z. Khaliq, S. U. Farooq, and K. D. Ashraf, "Artificial Intelligence in Software Testing : Impact, Problems, Challenges and Prospect," 2022, [Online]. Available: http://arxiv.org/abs/2201.05371

[14] D. Larkman, M. Mahammadian, and B. Balachandran, "General Application of a Decision Support Framework for Software Testing Using Artificial Intelligence," pp. 53–63, 2010.

[15] P. Srivastava and K. Tai-hoon, "Application of genetic algorithms in software testing," *Adv. Mach. Learn. Appl. Softw. Eng.*, no. November 2009, pp. 287–317, 2009, doi: 10.4018/978-1-59140-941-1.ch012.

[16] A. R. Lenz, A. Pozo, and S. R. Vergilio, "Engineering Applications of Arti fi cial Intelligence Linking software testing results with a machine learning approach," *Eng. Appl. Artif. Intell.*, vol. 26, no. 5–6, pp. 1631–1640, 2013, doi: 10.1016/j.engappai.2013.01.008.

[17] D. Chhillar and K. Sharma, "Proposed T-Model to cover 4S quality metrics based on empirical study of root cause of software failures," *Int. J. Electr. Comput. Eng.*, vol. 9, no. 2, p. 1122, 2019, doi: 10.11591/ijece.v9i2.pp1122-1130.

[18] J. Kahles, J. Torronen, T. Huuhtanen, and A. Jung, "Automating root cause analysis via machine learning in agile software testing environments," *Proc. - 2019 IEEE 12th Int. Conf. Softw. Testing, Verif. Validation, ICST 2019*, no. June, pp. 379–390, 2019, doi: 10.1109/ICST.2019.00047.

[19] J. Hu, W. Yi, N. W. Chen, Z. J. Gou, and W. Shuo, "Artificial neural network for automatic test oracles generation," *Proc. - Int. Conf. Comput. Sci. Softw. Eng. CSSE 2008*, vol. 2, no. 05, pp. 727–730, 2008, doi: 10.1109/CSSE.2008.774.

[20] Vineeta, A. Singhal, and A. Bansal, "Generation of test oracles using neural network and decision tree model," *Proc. 5th Int. Conf. Conflu. 2014 Next Gener. Inf. Technol. Summit*, pp. 313–318, 2014, doi: 10.1109/CONFLUENCE.2014.6949311.

[21] F. Wang, L. W. Yao, and J. H. Wu, "Intelligent test oracle construction for reactive systems without explicit specifications," *Proc. - IEEE 9th Int. Conf. Dependable, Auton. Secur. Comput. DASC 2011*, pp. 89–96, 2011, doi: 10.1109/DASC.2011.39.