Artificial Intelligence in Software Testing: A Systematic Review

Mahmudul Islam
Department of Computer Science and Engineering
Independent University, Bangladesh
mahmud@iub.edu.bd

Farhan Khan
Department of Computer Science and Engineering
Independent University, Bangladesh
farhankhan_262@yahoo.com

Sabrina Alam
Department of Computer Science and Engineering
Independent University, Bangladesh
sabrina.alam@iub.edu.bd

Mahady Hasan
Department of Computer Science and Engineering
Independent University, Bangladesh
mahady@iub.edu.bd

***Abstract*—Software testing is a crucial component of software development. With the increasing complexity of software systems, traditional manual testing methods are becoming less feasible. Artificial Intelligence (AI) has emerged as a promising approach to software testing in recent years. This systematic review study aims to provide the recent trend and the current state of software testing using AI. This study examines different types of approaches, techniques, and tools used in this area and assesses their effectiveness. The selected articles for this study have been extracted from different research databases using a search string. Initially, 90 articles were extracted from different research libraries. After gradual filtering in three different phases, 20 articles were selected for final review. Around 50 articles were studied to explore the use of AI in software testing and get an in-depth overview of it. The findings of this study suggest that various testing tasks can be automated successfully using AI, including Machine Learning (ML) and Deep Learning (DL), such as Test Case Generation, Defect Prediction, Test Case Prioritization, Metamorphic Testing, Android Testing, Test Case Validation, and White Box Testing. This study concludes that the integration of AI in software testing is simplifying software testing activities while improving overall performance. This study offers a comprehensive analysis of the utilization of AI techniques in different software testing activities.**

***Index Terms*—Software Testing, Artificial Intelligence, Machine Learning, Deep Learning, Test Automation, Systematic Literature Review**

I. Introduction

Software testing has a crucial role in software engineering as it is essential for ensuring the quality, performance, security, and reliability of software systems. By conducting testing, developers can identify and rectify any bugs, or defects in the software. It improves overall functionality and makes sure that the software satisfies customer needs and expectations. AI is a vast area, so in this study, we mainly investigate the subarea of AI which are ML and DL techniques in software testing. The field of software testing currently faces a number of challenges. As software systems grow increasingly complex, it becomes more challenging to manually test all possible scenarios. Also, traditional test automation approaches are time-consuming and complex to implement. Apart from that, keeping pace with agile development is also a challenge as it requires rapid testing. AI has the potential to address these challenges by offering optimized and effective testing strategies.

The aim of this study is to find the recent trends and the current state of the field of software testing automation using AI. This study examines the various methods, techniques, and tools utilized in this domain and evaluates their efficiency. The motivation for this study comes from the potential benefits of AI that can be offered in the field of software testing to improve the existing software testing practices. AI has the potential to automate the testing process and optimize testing strategies. AI can make software testing more efficient, effective, and accessible. Moreover, AI can address the shortage of skilled testers. Also, It can help to keep pace with the rapid development cycles of agile development methodologies. There are several challenges in software testing that can be solved using AI. Some of these issues include manually generating test cases, test optimization, test results analysis, etc. We tried to identify the recent trends in software testing using AI and came up with the following research questions which have been investigated in this research study.

RQ1: Does manual testing have drawbacks?
RQ2: Can integration of AI in software testing help to overcome the drawbacks of manual testing?
RQ3: What software testing tasks can be automated by AI?
RQ4: What techniques do researchers use to assess AI techniques when used in software testing?

In this research study, 90 articles or research studies have been screened from different research libraries. In three different phases using PRISMA guidelines, we came out with 20 research studies for final review. The contributions of this study are mentioned below.

- To identify recent trends in software testing using AI.
- To identify AI tools and techniques for automating software testing.
- To identify software testing activities automated by AI.

The remaining paper is structured in this way. Related works and the background of software testing and AI are discussed in sections 2 and 3 consecutively. The methodology of this systematic review, results, and conclusions are discussed in sections 4,5 and 6.

II. Related Works

They [1] proposed a deep learning model to rank test cases. In this work, they consider historical records of test case executions and based on that deep learning model rank test cases. They [2] conducted an empirical study on continuous integration testing. They found the strategy of reward function of Reinforcement learning improves the existing test case prioritization practices. They [3] developed a deep reinforcement learning technique for performing black box testing on Android apps. Their developed technique outperforms existing techniques in terms of fault identification. They [4] proposed a deep learning-based approach for prioritizing test cases from the interaction of humans with software applications. They showed that test case prioritization can be performed successfully from human interactions using their proposed model. They [5] presented an approach to generate input for the graphical user interface of software applications by only capturing screenshots of applications.

They [6] proposed an ML-based approach to predict metamorphic relations of scientific software using graph kernels. They concluded that features extracted from graphs help to achieve a good result. They [7] presented an approach to automate test oracle mechanism using ML. Their proposed approach captures historical usage data and based on that generates an oracle. They [8] detected metamorphic relations using graph kernels and support vector machines (SVM). They [9] analyzed software defect predictions using ML algorithms. They found that linear classifier performs well compared to other algorithms. They [10] proposed an improved CNN model to predict software defects and their proposed model outperformed existing models.

III. Software Testing & Artificial Intelligence

Software Testing is a process to evaluate the software and identify defects [11]. It is crucial for software to work or perform as per requirements but it is natural having bugs or defects in software. The bugs can be generated during development, bug fixing, feature addition, code refactoring, and even during software maintenance [12]. Therefore, it is crucial for the development team to test the software under different scenarios before releasing it to the client. There are different strategies and techniques for software testing. Based on the nature of the software it is decided which software testing technique should be used [13]. Software testing techniques are very tedious therefore, automation comes here to ease the process. How AI can automate software testing and why it is getting more acceptance than any other technique is discussed in this section. AI is a broad area that consists of various subareas, and ML is one of the most prominent and widely applied subareas within AI. Deep Learning (DL) and Machine Learning (ML) can be collectively referred to as Machine Learning (ML).

A. Software Testing Using Machine Learning

ML is a process where machines learn from data using algorithms and can further predict or make decisions based on the data [14]. The data-centric learning approach has made ML powerful and widely accepted in different areas including the software industry. Fig. 1 presents the general approach to applying ML techniques in software testing. There are different software testing activities such as defect prediction, test case generation, test case prioritization, test optimization, API testing, etc that can be done using ML [15].

Bug Prediction using ML: Bug prediction can be performed using ML. ML algorithms analyze software code and predict the likelihood of future bugs in the code. For performing bug prediction, ML models need to be trained on historical data from past software projects to identify patterns. Once the model is trained, then it can predict the likelihood of bugs occurring in new code [16]. They [17] used supervised ML algorithms to predict software faults based on historical data.

Test Case Generation using ML: In software development, Test case generation from the requirement specifications document is one of the significant challenges in software testing. Software test cases can be generated using ML. ML model needs to be trained on a set of data where a set of software features are considered as input and the corresponding test cases as output. Finally, the model uses training data to generate new test cases [18].

Test Case Prioritization using ML: Test case prioritization can be performed using ML. ML algorithms determine the most critical test cases to execute based on the likelihood of failure and the potential effect on the system. For prioritizing test cases, an ML model needs to be trained on a set of labeled data, where a set of software features is considered as input and the corresponding priority level of each test case as output. Finally, the model uses this training data to prioritize new test cases based on their predicted priority level [1].

IV. Method

A review is a systematic study that helps to identify the existing work, research questions improvement scope, and existing empirical studies [19]. In this study, 20 research studies have been reviewed from the past 7 years. these studies were collected from 6 different databases such as ScienceDirect, IEEE, SCITEPRESS, ACM, Wiley Online Library and MDPI.

A. Eligibility Criteria and Search String

Eligibility criteria for selecting articles for a systematic literature review include relevance to the research questions, publication time frame, language, publisher, and study design [20]. In this study, 20 articles were selected for final review out of 90 articles from the last 7 years, after filtering using PRISMA guidelines. Articles relevant to software testing using AI techniques such as ML and DL were chosen. In this process, we have used a search string which was formed to find articles related to the research study's area of interest. Boolean operators (AND, OR, NOT) have been used to combine and exclude keywords in the search query [21]. Our search string was

[("Software Testing" AND "Artificial Intelligence") AND ("Testing Automation Technique" OR "Machine Learning" OR "Deep Learning" OR "Black-box Testing" OR "Integration Testing" OR "Metamorphic Testing" OR "White Box Testing") NOT ("Manual Testing" OR "Adhoc testing")].

In addition to the search string approach, titles, keywords, abstracts and methods have been examined to find out relevant publications.

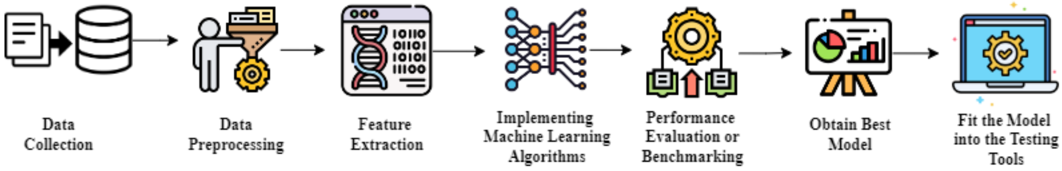

Fig. 1. A general approach to apply ML techniques in software testing

B. Data Screening and Extraction

Each paper examines different aspects of applications of ML techniques in software testing. In these studies, the authors applied different ML techniques, compared their performance in software testing, and came out with the best ML strategy to use in software testing. For collecting the research studies, PRSIMA [22] guidelines have been followed. The PRISMA flow diagram to select the articles for this systematic review study is presented in Fig. 2. In three different stages, the articles were screened. In the first stage, which is the identification stage, there were initially 90 articles in the database and 12 articles in the registers. In the second stage, articles were excluded due to being out of scope and poor quality. We read the title, keywords, abstract, and methods of each article to identify whether the article is relatable or not. Finally, 20 articles were included for final review. The inclusion and exclusion criteria used in this study to select the articles are presented in Fig. 3.

Area	Criteria for inclusion	Criteria for Exclusion
Article type	Research article	Book, Poster, Abstract
Searched keywords	Software testing, Artificial Intelligence, Machine learning, Testing automation, Test data generation, Blackbox testing, Whitebox testing	Keywords not included in the "Inclusion criteria"
Interest of area	Software testing, Software engineering, Artificial intelligence	Area excluding "Inclusion criteria"
Time period	2016 -2022	Before 2016

Fig. 3. Inclusion and Exclusion Criteria

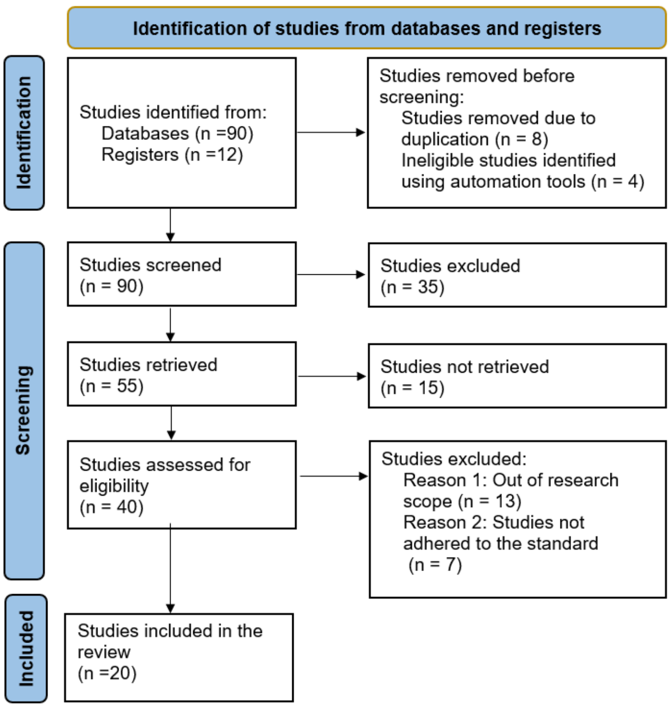

Fig. 2. PRISMA flow diagram used in this study for finding literature

Data extraction means the process of retrieving relevant data from various sources for a specific purpose, such as a literature review [23]. In the context of software testing using AI, data extraction may involve searching through academic journals, and conference proceedings to gather information on the latest developments and trends in software testing using AI. This information can then be used to summarise a comprehensive review of the current state of the field, identify gaps in existing knowledge, and provide insights into future directions for research and practice. Table I shows the details of the selected number of studies in different stages and their publishers.

V. Results

This section provides insights into state-of-the-art techniques and their effectiveness in improving the quality and efficiency of software testing using AI techniques. This study aims to provide a comprehensive synopsis of the existing research in this domain by analyzing a number of studies. 20 studies have been reviewed in the study and the details findings and analysis of these studies have been presented in Table II. We also investigated the answers to the research questions from the relevant research papers.

RQ1: Does manual testing have drawbacks?

Manual testing has several drawbacks. Some of the drawbacks of manual testing are it is time-consuming, it does not cover all possible scenarios and use cases, it is costly, it is susceptible to human errors and it can not reproduce test cases accurately [24]. ML techniques can help to overcome the mentioned drawbacks of manual testing. By leveraging the power of ML algorithms, the software testing process can be automated, and more accurate testing can be performed [25].

RQ2: Can integration of AI in software testing help to overcome the drawbacks of manual testing?

Integration of AI techniques in software testing can help to overcome the drawbacks of manual testing by improving the efficiency, accuracy, and effectiveness of the testing

TABLE I
SELECTED NUMBER OF RESEARCH STUDIES IN DIFFERENT STAGES

Publisher Name	First Stage: Identification	Second Stage: Screening	Third Stage: Included
IEEE	22	14	8
ACM	23	12	6
Science Direct	12	5	2
MDPI	18	4	2
Wiley	10	3	1
SCITEPRESS	5	2	1
Total	90	40	20

process. ML algorithms can be trained to automate repetitive testing tasks, which reduces the required effort for manual testing. This improves the efficiency of the software testing process and enables faster testing. ML algorithms can also analyze large amounts of data that help to identify defects in the software system. Identification of the defects improves the accuracy of the software testing. Apart from that, ML algorithms can generate test cases using historical data or existing code, and optimize the testing by prioritizing test cases [26].

RQ3: What software testing tasks can be automated by AI ?

ML techniques can automate different types of software testing tasks such as test results analysis, test case prioritization, defect prediction, test execution, test case evaluation, test case refinement, testing cost estimation, test oracle construction, identification of metamorphic relations, and test case generation [26]. Table III shows testing activities automated by ML techniques.

RQ4: What techniques do researchers use to assess AI techniques when used in software testing?

Researchers consider different performance matrices to assess ML algorithms when used in software testing. The performance matrices are cross-validation, accuracy, precision, recall, receiver operating characteristic (ROC) curve, area under the curve (AUC), and f1 score [27]. the details of performance matrices are described below.

Precision: Precision is a statistical measure that quantifies the ratio of true positive instances out of the total positive predictions made. [28].

Recall: Recall is a statistical indicator utilized to quantify the fraction of true positive outcomes within the entirety of actual positive instances [28].

ML algorithms have shown promising results in automating software testing tasks. Some of the promising algorithms are Neural networks, Decision trees, Support vector machines, and Random Forest.

VI. CONCLUSIONS

Software testing plays a key role in the development of software. However, as software systems become more complex, traditional manual testing methods are becoming less practical. There has been growing interest in leveraging AI techniques for software testing. This study explores the current state of the art of AI techniques in software testing. Also, this study examines various approaches, techniques, and tools employed in this field, assessing their effectiveness. The research articles selected for this review study were obtained from different research databases using a search string. The title, abstract, keywords, and methods of these articles were also checked manually. Initially, 90 articles were retrieved, and after rigorous filtering following the PRISMA guideline, 20 articles were chosen for final analysis.

This study finds that AI techniques can help in the automation of several software testing tasks. These tasks include Test Case Generation, Defect Prediction, Test Case Prioritization, Metamorphic Testing, Android Testing, Test Case Validation, and White Box Testing. The integration of AI techniques in software testing is shown to simplify software testing activities and enhance performance. In the future, incorporating AI techniques in different software testing activities will make it easier to perform testing activities. A limited number of studies have been examined in this study, which is a limitation. Conducting a review of a larger number of studies would provide the opportunity to gain deeper insights.

REFERENCES

[1] Sharif, A., Marijan, D. and Liaaen, M., 2021, September. DeepOrder: Deep learning for test case prioritization in continuous integration testing. In 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 525-534). IEEE.

[2] Yang, Y., Li, Z., He, L. and Zhao, R., 2020. A systematic study of reward for reinforcement learning based continuous integration testing. Journal of Systems and Software, 170, p.110787.

[3] Romdhana, A., Merlo, A., Ceccato, M. and Tonella, P., 2022. Deep reinforcement learning for black-box testing of android apps. ACM Transactions on Software Engineering and Methodology (TOSEM), 31(4), pp.1-29.

[4] Li, Y., Yang, Z., Guo, Y. and Chen, X., 2019, November. Humanoid: A deep learning-based approach to automated black-box android app testing. In 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 1070-1073). IEEE.

[5] YazdaniBanafsheDaragh, F. and Malek, S., 2021, November. Deep GUI: black-box GUI input generation with deep learning. In 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 905-916). IEEE.

[6] Kanewala, U., Bieman, J.M. and Ben-Hur, A., 2016. Predicting metamorphic relations for testing scientific software: a machine learning approach using graph kernels. Software testing, verification and reliability, 26(3), pp.245-269.

[7] Braga, R., Neto, P.S., Rabêlo, R., Santiago, J. and Souza, M., 2018, September. A machine learning approach to generate test oracles. In Proceedings of the XXXII Brazilian Symposium on Software Engineering (pp. 142-151).

[8] Nair, A., Meinke, K. and Eldh, S., 2019, August. Leveraging mutants for automatic prediction of metamorphic relations using machine learning. In Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation (pp. 1-6).

[9] Singh, P.D. and Chug, A., 2017, January. Software defect prediction analysis using machine learning algorithms. In 2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence (pp. 775-781). IEEE.

[10] Pan, C., Lu, M., Xu, B. and Gao, H., 2019. An improved CNN model for within-project software defect prediction. Applied Sciences, 9(10), p.2138.

[11] Myers, G.J., Sandler, C. and Badgett, T., 2011. The art of software testing. John Wiley & Sons.

[12] Tan, L., Liu, C., Li, Z., Wang, X., Zhou, Y. and Zhai, C., 2014. Bug characteristics in open source software. Empirical software engineering, 19, pp.1665-1705.

TABLE II
SUMMARY OF THE SELECTED STUDIES

SL	Source	Year	Publisher Name	Findings
1	[29]	2022	ACM	Authors proposed an approach utilizing Deep Reinforcement Learning (RL) for automating the exploration of Android apps. Authors developed a tool called ARES along with FATE that integrates with ARES.
2	[30]	2022	MDPI	This paper analyzed ML frameworks in the context of software automation and evaluated the performance of testing tools considering various factors. Accuracy or error rate, scope are important factors to determine the effectiveness of frameworks.
3	[31]	2022	Science Direct	This study investigates the efficacy of machine learning, data mining, and deep learning methodologies in predicting software faults. This investigation reveals that data mining and machine learning techniques are utilized more than deep learning techniques.
4	[32]	2022	ACM	This paper introduces Keeper, a novel testing tool. Keeper adopts a unique approach where it creates pseudo-inverse functions for ML APIs. Keeper significantly enhances branch coverage .
5	[33]	2021	IEEE	This study presents DeepOrder, a regression machine learning model based on deep learning techniques. DeepOrder can prioritize test cases and identify failed test cases when it considers various factors such as test case duration and execution status.
6	[34]	2021	Science Direct	This study investigated reward function and reward strategy within the context of continuous integration (CI) testing. The authors proposed three strategies in terms of the reward strategy. Proposed strategies showed promising results.
7	[5]	2021	IEEE	This paper introduces Deep GUI. Deep GUI utilizes deep learning techniques to create a model of valid GUI interactions, based solely on screenshots of applications.
8	[35]	2021	IEEE	This study finds that most ML libraries lack a high-quality unit test suite. Moreover, the study also discovers recurring trends in the unexamined code throughout the five assessed ML libraries.
9	[36]	2021	IEEE	This study presents a deep learning approach to predict the validity of test inputs for RESTful APIs. The proposed network achieved 97% accuracy for the new APIs.
10	[37]	2019	IEEE	This paper introduces Humanoid, a deep learning approach for generating GUI test inputs by leveraging knowledge gained from human interactions. It learns from traces of interactions generated by humans, enabling the automatic prioritization of test inputs based on their perceived importance to users.
11	[38]	2019	ACM	This study finds equivalent mutants are effective for augmenting data and improving the detection rate of metamorphic relations.
12	[39]	2019	MDPI	This study introduces an enhanced CNN model specifically designed to improve the learning of semantic representations from source-code. This study also showed enhancements of the global pattern capture capability of the models which improve the model's generalization performance.
13	[40]	2019	IEEE	This study used three supervised machine learning algorithms for predicting software bugs. To enhance the accuracy of models, random forest ensemble classifiers have been used. The developed models effectively work for various scenarios.
14	[41]	2019	IEEE	This study finds ML algorithms have predominantly been employed in different areas of software testing. Test case generation, evaluation, test oracle construction, and cost predicton for testing activitires can be performed using ML.
15	[42]	2018	ACM	This study presents an approach for automating the test oracle mechanism in software using machine learning (ML). By incorporating a captured component into the application, historical usage data have been gathered. These data later generate an appropriate oracle.
16	[43]	2018	SCITE PRESS	This paper describes a tool that generates test data for programs. The tool operates by clustering input data from a corpus folder and creating generative models for each cluster. These models are recurrent neural networks.
17	[44]	2018	ACM	This paper introduces a methodology called DaOBML, which offers tool support to enhance the quality of environmental models that generate complex artifacts like images or plots. In this study, among six ML algorithms, ANN shows the best performance.
18	[45]	2017	ACM	This study introduces DeepXplore, an innovative whitebox system designed to systematically test DL systems and detect faulty behaviors. DeepXplore can solve joint optimization problems.
19	[46]	2016	Wiley Online Library	This study, proposed a ML approach that can predict metamorphic relations in software programs. To achieve this, authors utilized a graph-based representation of the program.
20	[47]	2016	IEEE	This study proposed an approach for prioritizing test cases in manual testing. The proposed approach considers black-box metadata, including test case history. SVM Rank ML algorithm is used in this study.

TABLE III
TESTING ACTIVITIES AUTOMATED BY ML TECHNIQUES

Software Testing Activity	Total No. of Studies
Test Case Generation	4
Defect Prediction	3
Test Case Prioritization	3
Metamorphic Testing	2
Android Testing	2
Test Case Validation	1
White Box Testing	1

[13] Jamil, M.A., Arif, M., Abubakar, N.S.A. and Ahmad, A., 2016, November. Software testing techniques: A literature review. In 2016 6th international conference on information and communication technology for the Muslim world (ICT4M) (pp. 177-182). IEEE.

[14] Jordan, M.I. and Mitchell, T.M., 2015. Machine learning: Trends, perspectives, and prospects. Science, 349(6245), pp.255-260.

[15] Zhou, Z.H., 2021. Machine learning. Springer Nature.

[16] Efendioglu, M., Sen, A. and Koroglu, Y., 2018. Bug prediction of systemc models using machine learning. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 38(3), pp.419-429.

[17] Hammouri, A., Hammad, M., Alnabhan, M. and Alsarayrah, F., 2018. Software bug prediction using machine learning approach. International journal of advanced computer science and applications, 9(2).

[18] Zhang, D., 2006, November. Machine learning in value-based software test data generation. In 2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06) (pp. 732-736). IEEE.

[19] Kitchenham, B. and Brereton, P., 2013. A systematic review of systematic review process research in software engineering. Information and software technology, 55(12), pp.2049-2075.

[20] Papaioannou, D., Sutton, A. and Booth, A., 2016. Systematic approaches to a successful literature review. Systematic approaches to a successful literature review, pp.1-336.

[21] Mohamed Shaffril, H.A., Samsuddin, S.F. and Abu Samah, A., 2021. The ABC of systematic literature review: the basic methodological guidance for beginners. Quality & Quantity, 55, pp.1319-1346.

[22] Page, M.J., McKenzie, J.E., Bossuyt, P.M., Boutron, I., Hoffmann, T.C., Mulrow, C.D., Shamseer, L., Tetzlaff, J.M., Akl, E.A., Brennan, S.E. and Chou, R., 2021. The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. International journal of surgery, 88, p.105906.

[23] Jonnalagadda, S.R., Goyal, P. and Huffman, M.D., 2015. Automating data extraction in systematic reviews: a systematic review. Systematic reviews, 4(1), pp.1-16.

[24] Leitner, A., Ciupa, I., Meyer, B. and Howard, M., 2007, January. Reconciling manual and automated testing: The autotest experience. In 2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07) (pp. 261a-261a). IEEE.

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

[26] Durelli, V.H., Durelli, R.S., Borges, S.S., Endo, A.T., Eler, M.M., Dias, D.R. and Guimarães, M.P., 2019. Machine learning applied to software testing: A systematic mapping study. IEEE Transactions on Reliability, 68(3), pp.1189-1212.

[27] Song, Q., Guo, Y. and Shepperd, M., 2018. A comprehensive investigation of the role of imbalanced learning for software defect prediction. IEEE Transactions on Software Engineering, 45(12), pp.1253-1269.

[28] Tatbul, N., Lee, T.J., Zdonik, S., Alam, M. and Gottschlich, J., 2018. Precision and recall for time series. Advances in neural information processing systems, 31.

[29] Romdhana, A., Merlo, A., Ceccato, M. and Tonella, P., 2022. Deep reinforcement learning for black-box testing of android apps. ACM Transactions on Software Engineering and Methodology (TOSEM), 31(4), pp.1-29.

[30] Fatima, S., Mansoor, B., Ovais, L., Sadruddin, S.A. and Hashmi, S.A., 2022. Automated Testing with Machine Learning Frameworks: A Critical Analysis. Engineering Proceedings, 20(1), p.12.

[31] Batool, I. and Khan, T.A., 2022. Software fault prediction using data mining, machine learning and deep learning techniques: A systematic literature review. Computers and Electrical Engineering, 100, p.107886.

[32] Wan, C., Liu, S., Xie, S., Liu, Y., Hoffmann, H., Maire, M. and Lu, S., 2022, May. Automated testing of software that uses machine learning apis. In Proceedings of the 44th International Conference on Software Engineering (pp. 212-224).

[33] Sharif, A., Marijan, D. and Liaaen, M., 2021, September. DeepOrder: Deep learning for test case prioritization in continuous integration testing. In 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 525-534). IEEE.

[34] Yang, Y., Li, Z., He, L. and Zhao, R., 2020. A systematic study of reward for reinforcement learning based continuous integration testing. Journal of Systems and Software, 170, p.110787.

[35] Wang, S., Shrestha, N., Subburaman, A.K., Wang, J., Wei, M. and Nagappan, N., 2021, May. Automatic unit test generation for machine learning libraries: How far are we?. In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE) (pp. 1548-1560). IEEE.

[36] Mirabella, A.G., Martin-Lopez, A., Segura, S., Valencia-Cabrera, L. and Ruiz-Cortés, A., 2021, June. Deep learning-based prediction of test input validity for RESTful APIs. In 2021 IEEE/ACM Third International Workshop on Deep Learning for Testing and Testing for Deep Learning (DeepTest) (pp. 9-16). IEEE.

[37] Li, Y., Yang, Z., Guo, Y. and Chen, X., 2019, November. Humanoid: A deep learning-based approach to automated black-box android app testing. In 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 1070-1073). IEEE.

[38] Nair, A., Meinke, K. and Eldh, S., 2019, August. Leveraging mutants for automatic prediction of metamorphic relations using machine learning. In Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation (pp. 1-6).

[39] Pan, C., Lu, M., Xu, B. and Gao, H., 2019. An improved CNN model for within-project software defect prediction. Applied Sciences, 9(10), p.2138.

[40] Immaculate, S.D., Begam, M.F. and Floramary, M., 2019, March. Software bug prediction using supervised machine learning algorithms. In 2019 International conference on data science and communication (IconDSC) (pp. 1-7). IEEE.

[41] Durelli, V.H., Durelli, R.S., Borges, S.S., Endo, A.T., Eler, M.M., Dias, D.R. and Guimarães, M.P., 2019. Machine learning applied to software testing: A systematic mapping study. IEEE Transactions on Reliability, 68(3), pp.1189-1212.

[42] Braga, R., Neto, P.S., Rabêlo, R., Santiago, J. and Souza, M., 2018, September. A machine learning approach to generate test oracles. In Proceedings of the XXXII Brazilian Symposium on Software Engineering (pp. 142-151).

[43] Paduraru, C. and Melemciuc, M.C., 2018, July. An Automatic Test Data Generation Tool using Machine Learning. In ICSOFT (pp. 506-515).

[44] de Santiago, V.A., da Silva, L.A.R. and de Andrade Neto, P.R., 2018, September. Testing environmental models supported by machine learning. In Proceedings of the III Brazilian Symposium on Systematic and Automated Software Testing (pp. 3-12).

[45] Pei, K., Cao, Y., Yang, J. and Jana, S., 2017, October. Deepxplore: Automated whitebox testing of deep learning systems. In proceedings of the 26th Symposium on Operating Systems Principles (pp. 1-18).

[46] Kanewala, U., Bieman, J.M. and Ben-Hur, A., 2016. Predicting metamorphic relations for testing scientific software: a machine learning approach using graph kernels. Software testing, verification and reliability, 26(3), pp.245-269.

[47] Lachmann, R., Schulze, S., Nieke, M., Seidl, C. and Schaefer, I., 2016, December. System-level test case prioritization using machine learning. In 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA) (pp. 361-368). IEEE.