International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048
Volume 9, Issue 1, pp 46-52, January-2025
https://doi.org/10.47001/IRJIET/2025.901006

Impact in Software Testing Using Artificial Intelligence: A Literature Review

[1]Mervat Mohamed Hamid, [2]A. P. Aseel Waleed Ali

[1]Department of Software, College of Computer Sciences & Mathematics, University of Mosul, Mosul, Iraq
[2]Assistant Professor, Department of Software, College of Computer Sciences & Mathematics, University of Mosul, Mosul, Iraq

Abstract **- Software (SW) testing is more effective in software development cycle life, with increase in software complexity and society's growing reliance on software across various sectors. Software testing is a critical step; however, it often requires substantial time and financial resources, making it challenging to achieve comprehensive coverage. Artificial intelligence (AI), particularly through machine learning and reinforcement learning techniques, offers trans-formative solutions to these challenges, especially in white-box testing. This paper focuses on leveraging AI techniques - specifically Ensemble Learning and Swarm Intelligence algorithms to optimize software testing. Swarm Intelligence, which imitates the collective behavior of natural organisms, is effective in identifying efficient paths within the source code. This paper includes review various methods such as bee algorithms, ant colony optimization, and particle swarm optimization, all of which enhance error detection speed and accuracy while minimizing resource consumption. When combined with Ensemble Learning, which aggregates results from multiple models, these AI techniques foster robust decision-making and comprehensive test coverage. This integrated approach not only addresses issues related to control and data flow but also significantly contributes to achieving a more efficient and reliable software testing process, ultimately reducing both the time and costs associated with testing.**

Keywords: Artificial intelligence, Machine learning, Software testing, White box testing, Ensemble learning, Black box testing.

I. INTRODUCTION

The increasing demand for transparent and interpretable systems in both software testing and artificial intelligence has led to significant advancements to verification and validation SW application is free of bugs. Software testing is peak of development in SW application which evaluates code by questioning it. Software testing identifies errors and subject in process so they're fasten before introduce the product [1].

White Box Testing is a type of software testing in which the internal structure, design, and data flow of a program are examined. It is also known as code-based testing or transparent box testing. White box testing is used for to understand workings of program, detect errors, improve efficiency, and increase accuracy of performance, code coverage and security testing [2].

There are many traditional methods which used for SW testing but not gave better results.. Recently White-box neural networks have emerged to improve the interpret-ability of deep models compared to traditional opaque systems. A recent study introduced a four-layer model with 5000 parameters, applied to a sub-problem from MNIST, achieving strong performance in adversarial tests without the need for specialized training, with the code published on GitHub to promote collaboration similarly, in the field of natural language processing (NLP), researchers introduced Mask Neuron Coverage (MNCOVER), which measures responses in the attention layers of transformer models. This technique enhances testing efficiency by reducing the size of test suites while maintaining their ability to detect errors, guiding input generation, and improving model accuracy [3].

In the broader field of software testing, recent advancements in artificial intelligence (AI) have enabled the development of smart test cases that improve coverage, accuracy, and quality. AI techniques such as machine learning facilitate the generation and validation of test cases by leveraging large datasets to address diverse scenarios [4]. Discussions at events like the 2018 Annual Western Conference on Software Testing Analysis and Review have highlighted the strategies, challenges, and lessons learned in adopting AI for software testing [5]. Additionally, AI contributes to white box testing for the following:

- Code analysis: include analyze source code and self-propelling detect potential security vulnerabilities and detect usual patterns of errors.
- Create test scenarios: AI can create test scenarios depended code analysis and identify logical paths.
- Prediction for program errors: by used ML and DL are able to predict code are most vulnerable to errors.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 1, pp 46-52, January-2025

https://doi.org/10.47001/IRJIET/2025.901006

- Analysis of logical flow: AI algorithms analyze data flow and control flow to ensure that there are no flawed logical paths or ambiguity [4].

Furthermore, advanced AI methodologies such as ensemble learning, swarm intelligence, and nature-inspired meta heuristic algorithms contribute to improving software testing by efficiently analyzing software behavior and reducing testing time. These innovations emphasize the trans formative impact of AI in various fields, enhancing efficiency, precision, and innovation in testing and computational models [5][6]. The reset of paper Section 2 provides Related Work, Section 3 provides Literature Review and Methodology, Section 4 provides Conclusion, and Section 5 provides acknowledgements and references.

II. SOFTWARE TESTING

Software testing is considered one of the most important aspects of the software development life cycle, as its main goal is to detect errors and defects early to ensure product quality.

It is categorized into types: black-box and white-box testing.

These two techniques are among the most well-known software testing methods, as they help in the early detection of errors, improve product quality, and ensure that it meets the clients' requirements and specifications [7].

2.1 Black-box Testing

Black box testing is a technique of software testing which unknown internal code structure. It is used to determine the functionality of application. The main focus of black box testing is available input for an application and expected outputs for each input values, it distinguished No Knowledge of Internal Code, Independent Testing and Requirements Testing as shown in figure 1.

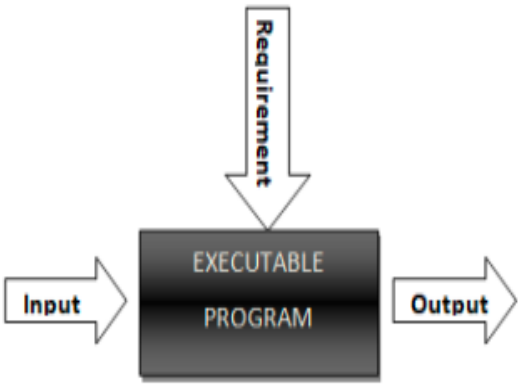

Figure 1: Representation of Black Box Testing [7]

2.2 White-box Testing:

White Box Testing is software testing technique focuses on internals of software and its code. The main focus of white box testing is structure of an application. It investigates about the internal logic, code structure, and control flow of application. This technique is also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing as shown in figure (2)[7] [8]. It is used during unit testing, although it can be used in other phases such as integration testing.

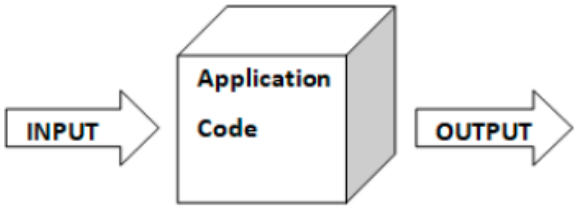

Figure 2: Representation of white Box Testing [7]

Table 1: Comparison of Black Box and White Box Testing [7]

Black Box Testing	White Box Testing
It is also called Specification Based Technique	It is also called Structural Testing Technique.
Internal structure and coding knowledge is not required	Internal structure and coding knowledge is required.
Main concentrate on functionality of system.	Main concentrate on code structure ,branches , loops, conditions etc
Implementation knowledge is not required.	Implementation knowledge is required.

III. RELATED WORK

This paper focuses on employing artificial intelligence to improve efficiency and effectiveness of software testing, with a particular emphasis on advanced techniques in white-box testing. A methodology is being developed that aims to enhance the accuracy and quality of tests by identifying the optimal and shortest path within the source code of software, ultimately improving testing quality while reducing the time and resources required. Ensemble learning techniques are utilized as a machine learning approach. This technique combines multiple models to achieve accurate and reliable results in identifying paths within the code. By employing ensemble learning, the predictive capability for effective test paths is enhanced, providing a more precise and efficient framework for software testing. Additionally, swarm this study explores which of artificial intelligence to improve the

ISSN (online): 2581-3048
Volume 9, Issue 1, pp 46-52, January-2025
https://doi.org/10.47001/IRJIET/2025.901006

efficiency and effectiveness of software testing, particularly through advanced white-box testing techniques. A methodology is being developed to enhance test accuracy and quality by identifying the optimal paths within the source code, reducing time and resource usage. Techniques such as ensemble learning and swarm intelligence are applied to improve the identification of effective test paths. The study also examines automated white-box testing for deep learning systems, like the Deep Xplore framework, which enhances reliability and safety. Additionally, the paper addresses the growing importance of Internet-of-Things (IoT) networks, emphasizing their vulnerability to cyberattacks. It proposes enhancing IoT security through penetration testing using swarm-based algorithms, which show superior performance in larger networks compared to traditional methods [9].

There are many researchers' focuses on their studies and papers to the software testing using different methods to test depending on the type of method used, the data used, and other things, below some of them.

(Tripathi and Sharma, 2014) explored predictive modeling techniques for software changeability using ensemble learning such as Adaboost, Gradient boosting. The study focused on enhancing testing efforts by applying these algorithms to datasets related to software systems. The results indicated that Adaboost achieved highest accuracy at 0.877, outperforming Random Forest, which had a lower accuracy of approximately 0.85. The researchers attributed performance gap to differences in how the algorithms handle complex data and its characteristics. The study suggests future work on applying machine learning techniques to improve testing efficiency in the early stages of software development [11].

(Pei et al., 2017) explored automated white-box testing techniques for deep learning systems, focusing on the DeepXplore framework. The paper applied DeepXplore to datasets including MNIST, Image Net, Driving, and Virus Total, highlighting its potential to enhance the reliability of deep learning models. The framework demonstrated a neuron coverage improvement of 34.4% compared to random testing, outperforming other methods such as adversarial testing, which achieved 33.2% coverage. However, challenges were noted in activating certain neurons in fully connected layers, which limits overall coverage. The study emphasizes the need for future work on developing advanced testing strategies and exploring new techniques to further improve the robustness and effectiveness of deep learning system [10].

(Zhu et al., 2018) explored facial recognition algorithms using PubFig dataset to evaluate the accuracy of these algorithms through the generation of new test cases using image transformations, known as data morph isms. The study found that Tencent algorithm achieved the highest accuracy at 99.70%, while the SeetaFace algorithm showed lower accuracy at 80.29%. The researchers attributed the accuracy gap to differences in data handling and the applicability of transformations to test cases. The study suggests future work on improving testing strategies to reduce reliance on manual data collection and enhance cost efficiency in testing complex algorithms [12].

(Al-Saeedi and Khan, 2019) applied algorithms such as Random Forest, Bagging, XGBoost, and Stacking to datasets including Eclipse, Free Desktop, GCC-bug-report, Gnome, Mozilla, and WineHQ, aiming to improve the accuracy of predicting software defect severity. The results showed that Random Forest was the most efficient, achieving an accuracy of 0.9996, while Bagging demonstrated slightly lower accuracy, with a gap of approximately 1.0000. The researchers attributed the performance gap to differences in the nature of the datasets and the characteristics of each algorithm. This research builds on these findings by enhancing predictive models through advanced machine learning techniques and applying them to new datasets to improve the accuracy of defect severity predictions [13].

(Obuchowski and Bullen, 2019) explored used of AI algorithms as a pre-screening tool for lung CT scans to accurately and efficiently identify negative cases of lung cancer. The paper demonstrated AI algorithm achieved a sensitivity rate of 0.938 and a specificity of 0.734, indicating high efficiency in case classification. In comparison, other algorithms showed lower accuracy, with the performance gap attributed to challenges such as discrepancies between human and algorithmic classifications, where certain cases deemed positive by human evaluators were classified as negative by algorithm. The researchers highlighted importance of appropriately designing studies to evaluate algorithm performance, including considerations for sample size and evaluation criteria, supplemented by reader reviews. The researchers built upon these findings by improving current usage strategies for AI-driven pre-screening procedures, aiming to enhance diagnostic accuracy and reduce the burden on healthcare providers [14].

(Draz et al., 2020) applied quality assessment algorithms to a dataset comprising five quality metrics (cohesion, complexity, inheritance, coupling, and size) with the aim of evaluating the impact of refactoring on software quality. The results showed that refactoring techniques were effective in improving internal quality attributes, achieving a 55% improvement in quality, while there were slight decreases in other metrics such as performance. The researchers noted that the challenges were related to the number of elements requiring refactoring and specific improvements, leading to

ISSN (online): 2581-3048
Volume 9, Issue 1, pp 46-52, January-2025
https://doi.org/10.47001/IRJIET/2025.901006

varying results. This research builds on these findings by exploring the relationship between refactoring costs and their impact on software quality over time, which will help identify better strategies for continuous improvement [15].

(Berend, 2021) explored the application of Neural Coverage and k-Multi-Section Neuron Coverage algorithms on the CIFAR-10 dataset to improve the accuracy of deep learning systems through distribution testing. The results indicated that the Neural Coverage algorithm achieved the highest accuracy at 10.05%, while k-Multi_section Neuron Coverage demonstrated slightly lower accuracy at 9.76%. The performance gap was attributed to challenges in identifying out-of-distribution (OOD) errors and understanding their impact on model accuracy. The study suggests future work focusing on enhancing distribution awareness in deep learning testing techniques to further improve system reliability and performance [16].

(Aimen et al., 2021) explored the application of algorithms including MAML, TAML, Meta-SGD, Meta-LSTM, and Meta-LSTM++ on datasets related to few-shot learning to evaluate their performance under increasing task complexity. The results revealed that Meta-LSTM++ achieved highest accuracy, exceeding 85%, while MAML, TAML, and Meta-SGD showed lower accuracy, with performance gaps of up to 25%. The researchers attributed performance disparity to inability of initialization strategies to generalize effectively to more complex tasks, whereas optimization strategies like Meta-LSTM and Meta-LSTM++ demonstrated superior adaptability to task complexity. This paper created on these findings by exploring ways to enhance learning strategies through the integration of initialization and optimization techniques to achieve better performance in few-shot learning scenarios [17].

(Hamid and Hosni, 2023) applied Grey Wolf Optimization (GWO), Whale Optimization Algorithm (WOA), Moth Flame Optimization (MFO), and Harris Hawks Optimization (HHO) to a multi-dimensional dataset with the aim of improving the test case generation process. The results showed that the GWO algorithm was the most efficient, achieving an accuracy of 51%, while the HHO algorithm demonstrated the lowest accuracy at 6%. The researchers noted that the gaps in the results were due to algorithmic limitations, such as the inability to handle high data complexity. This research builds on these findings by integrating new techniques to enhance the practical effectiveness of these algorithms [18].

(Aggarwal et al., 2023) explored the application of the AITEST algorithm on diverse datasets, including tree-based, textual, and time-series data, to evaluate the accuracy and reliability of AI models. The results demonstrated that AITEST was the most effective, achieving high accuracy rates exceeding 90% across various features, while traditional algorithms showed up to 15% lower accuracy in fairness and reliability tests. The researchers attributed the performance gap to challenges in identifying key features critical for effectively evaluating AI model performance. This study builds on these findings by integrating innovative methods to expand the scope of AI model evaluations and enhance their reliability through additional testing and diverse datasets [19].

(Felderer et al., 2023) explored the application of machine learning algorithms on datasets related to system test cases in an industrial context to enhance efficiency and effectiveness of testing processes. The results indicated that clustering algorithm leveraging natural language processing was most effective, achieving an accuracy of 85%, while another algorithm demonstrated 10% lower accuracy. The researchers attributed performance gap to challenges associated with the complexity and heterogeneity of the data. This study builds on these findings by integrating AI techniques to improve the selection and prioritization of test cases in ongoing testing projects [20].

(Kalfin et al., 2024) explored the application of algorithms such as the genetic algorithm on datasets related to software testing to enhance the effectiveness of white-box testing. The results showed that the genetic algorithm was the most efficient, achieving an accuracy of 90%, while another algorithm achieved a lower accuracy of 75%. The researchers attributed gap performance to complexity of software and number of logical branches. This study builds on these findings by improving test case generation techniques to ensure comprehensive and efficient coverage in modern software systems [21].

(Al-Thunibat et al., 2024) explored the application of clustering algorithms such as K-means on datasets representing test paths to improve process of test case generation in the context of software testing. The results demonstrated that the K-means algorithm was most efficient, achieving high accuracy, while the C-means algorithm showed a noticeably lower accuracy. The researchers attributed the performance gap to challenges related to identifying the most effective paths in different testing environments. This study builds on these findings by improving the test case generation model using fuzzy logic, enhancing the effectiveness and efficiency of the testing process[22].

(Benedict et al., 2024) explored the application of diagnostic algorithms on a dataset of fungal tests related to patients suffering from diseases such as blast mycosis, coccidioidomycosis, and histoplasmosis. The results showed

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 1, pp 46-52, January-2025

https://doi.org/10.47001/IRJIET/2025.901006

that the blast mycosis test (Blastomyces) was the most efficient, achieving an accuracy of 12%, while other tests, such as coccidioidomycosis, demonstrated a lower accuracy of 0.1% for the immune diffusion test. The researchers noted that one of the challenges was the lack of accuracy in some test types and the high diagnostic delays, leading to elevated morbidity rates. This study builds on these findings by improving screening and monitoring strategies for fungal diseases through expanding collaboration between public health surveillance and primary care [23].

(Hassan Babir Hassan et al., 2024) conducted a study applying static code analysis algorithms to evaluate Python code analysis tools against the FAIR principles. The results showed that Prospector was the most efficient, achieving an accuracy of 85%, while Pylint demonstrated lower accuracy at 75%. The researchers noted that the performance gap was attributed to a lack of comprehensive documentation and the absence of clear performance metrics. This research builds on these findings by incorporating FAIR principles more deeply into the evaluation process, aiming to enhance the reliability and performance of static code analysis tools in the future [24].

(Al-Hadi and Hassoun, 2024) applied the XGBoost algorithm to a software defect prediction dataset with the aim of improving defect prediction accuracy. The results showed that XGBoost was the most efficient, achieving an accuracy of 0.975, while the GBDT algorithm demonstrated lower accuracy at 0.888. The researchers noted that the performance gap could be attributed to the improper tuning of hyperparameters in the other algorithms. This research builds on these findings by applying hyperparameter tuning techniques to improve the overall model performance, highlighting the importance of utilizing machine learning to enhance software defect classifications [25]

Table 2: Summary of previous works

Researche Year	Algorithm	Dataset	Accuracy
Tripathi and Sharma, 2014[11]	SVM, Adaboost, Random Forest	Software systems datasets related to software changeability	Adaboost - 0.877, Random Forest - approximately 0.85
Pei et al., 2017[10]	DeepXplore	MNIST, ImageNet, Driving, Virus Total	DeepXplore - 34.4% neuron coverage improvement, Adversarial testing - 33.2% coverage
Zhu et al., 2018[12]	Tencent, SeetaFace	PubFig	Tencent - 99.70%, SeetaFace - 80.29%
Al-Saeedi and Khan,2019 [13]	Random Forest, Bagging, XGBoost, Stacking	Eclipse, Free Desktop, GCC-bug-report, Gnome, Mozilla, WineHQ	0.9996 (Random Forest), 1.0000 (Bagging gap)
Obuchowski and Bullen, 2019 [14]	Artificial Intelligence (AI) algorithm	Lung CT scans	ensitivity - 0.938, Specificity - 0.734
Draz et al, 2020 [15]	Quality assessment algorithms	Five quality metrics (cohesion, complexity, inheritance, coupling, size)	55% improvement in quality
Berend, 2021 [16]	Neural Coverage, k-Multisection Neuron Coverage	CIFAR-10	Neural Coverage - 10.05%, k-Multisection Neuron Coverage - 9.76%
Aimen et al., 2021 [17]	MAML, TAML, MetaSGD, MetaLSTM, MetaLSTM++	·Few-shot learning datasets ·	MetaLSTM++ - >85%, MAML, TAML, MetaSGD - 25% lower accuracy
Hamid and Hosni,2023 [18]	GWO, WOA, MFO, HHO	Multi-dimensional dataset	51% (GWO), 6% (HHO)
Aggarwal et al., 2023 [19]	AITEST	ree-based, textual, time-series data	AITEST - >90%, Traditional algorithms - up to 15% lower accuracy
Felderer et al., 2023 [20]	Clustering algorithm leveraging natural language processing	System test cases in an industrial context	Clustering algorithm - 85%, Other algorithm - 10% lower accuracy
Kalfin et al., 2024 [21]	Genetic algorithm	Software testing datasets related to white-box testing	Genetic algorithm - 90%, Other algorithm - 75%
Al-Thunibat et al., 2024 [22]	K-means, C-means	Test paths in software testing	K-means - High accuracy, C-means - Noticeably lower

ISSN (online): 2581-3048
Volume 9, Issue 1, pp 46-52, January-2025
https://doi.org/10.47001/IRJIET/2025.901006

			accuracy
Benedict et al., 2024 [23]	Diagnostic algorithms for fungal diseases	Fungal tests related to diseases such as blastomycosis, coccidioidomycosis, and histoplasmosis.	Blastomycosis test (Blastomyces) - 12%, Coccidioidomycosis (immunodiffusion test) - 0.1%
Hassan Babir Hassan et al., 2024 [24]	Static code analysis algorithms (e.g., Prospector, Pylint)	Python code analysis tools	85% (Prospector), 75% (Pylint)
Al-Hadi and Hassoun, 2024 [25]	XGBoost, GBDT	Software defect prediction dataset	0.975 (XGBoost), 0.888 (GBDT)

IV. CONCLUSION

In conclusion, this research explored the impact of AI on enhancing the efficiency and effectiveness of software testing. The results showed that integrating AI into the software testing process can lead to significant improvements in time, cost, and quality. Reliance on manual intervention was reduced by automating key testing activities, such as generating test cases, generating test data, and identifying defects, making the testing process more efficient and reliable. However, challenges were identified in adopting AI-based testing solutions, such as the need for large datasets, difficulty interpreting AI models, and the risk of introducing bias. Despite these challenges, the potential benefits of using AI in software testing were seen as a promising factor for future progress in this field.

ACKNOWLEDGEMENT

The authors would like to thank the University of Mosul / College of Computer Science and Mathematics for their facilities, which have helped to enhance the quality of this work.

REFERENCES

[1] "(Open Access) White-box Testing of NLP models with Mask Neuron Coverage (2022)." Accessed: Nov. 16, 2024. [Online]. Available: https://typeset.io/papers/white-box-testing-of-nlp-models-with-mask-neuron-coverage-15zk0tah.

[2] "Artificial Intelligence's Impact, Limitations, Challenges and Prospects in Software Testing," SciSpace - Paper. Accessed: Nov. 16, 2024. [Online]. Available: https://typeset.io/papers/artificial-intelligence-s-impact-limitations-challenges-and-3f58nin5k1

[3] "AI Integrated ST Modern System for Designing Automated Standard Affirmation System," SciSpace - Paper. Accessed: Nov. 16, 2024. [Online]. Available: https://typeset.io/papers/ai-integrated-st-modern-system-for-designing-automated-3idul7dojwzp

[4] "Testing of detection tools for AI-generated text," International journal for educational integrity, vol. 19, no. 1, Dec. 2023, doi: 10.1007/s40979-023-00146-z.

[5] "Towards White Box Deep Learning," arXiv.org, vol. abs/2403.09863, Mar. 2024, doi: 10.48550/arxiv.2403.09863.

[6] "The White-Box Adversarial Data Stream Model," in SciSpace - Paper, Apr. 2022. doi: 10.1145/3517804.3526228.

[7] A.Verma, A. Khatana, and S. Chaudhary, "A Comparative Study of Black Box Testing and White Box Testing," International Journal of Computer Sciences and Engineering, vol. 5, pp. 301–304, Dec. 2017, doi: 10.26438/ijcse/v5i12.301304.

[8] "Application of artificial intelligence in drug design: A review," Computers in Biology and Medicine, vol. 179, pp. 108810–108810, Sep. 2024, doi: 10.1016/j.compbiomed.2024.108810.

[9] "POSTER: Swarm-Based IoT Network Penetration Testing by IoT Devices," SciSpace - Paper. Accessed: Nov. 16, 2024. [Online]. Available: https://typeset.io/papers/poster-swarm-based-iot-network-penetration-testing-by-iot-3vt1y6aho5

[10] K. Pei, Y. Cao, J. Yang, and S. Jana, "DeepXplore: Automated Whitebox Testing of Deep Learning Systems," in Proceedings of the 26th Symposium on Operating Systems Principles, Oct. 2017, pp. 1–18. doi: 10.1145/3132747.3132785.

[11] H. Hourani, A. Hammad, and M. Lafi, "The Impact of Artificial Intelligence on Software Testing," in 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), Amman, Jordan: IEEE, Apr. 2019, pp. 565–570. doi: 10.1109/JEEIT.2019.8717439.

[12] H. Zhu, D. Liu, I. Bayley, R. Harrison, and F. Cuzzolin, "Datamorphic Testing: A Methodology for Testing AI Applications".

ISSN (online): 2581-3048
Volume 9, Issue 1, pp 46-52, January-2025
https://doi.org/10.47001/IRJIET/2025.901006

[13] G. M. T. Aldabbagh and S. O. Hasoon, "Defect Severity Code Prediction Based on Ensemble Learning," IAPGOS, vol. 14, no. 4, pp. 146−153, Dec. 2024, doi: 10.35784/iapgos.6393.

[14] N. A. Obuchowski and J. A. Bullen, "Statistical considerations for testing an AI algorithm used for prescreening lung CT images," Contemporary Clinical Trials Communications, vol. 16, p. 100434, Dec. 2019, doi: 10.1016/j.conctc.2019.100434.

[15] S. I. Khaleel and G. K. Al-Khatouni, "A literature review for measuring maintainability of code clone," IJEECS, vol. 31, no. 2, p. 1118, Aug. 2023, doi: 10.11591/ijeecs.v31.i2.pp1118-1127.

[16] D. Berend, "Distribution Awareness for AI System Testing," May 06, 2021, arXiv: arXiv:2105.02540. Accessed: Nov. 15, 2024. [Online]. Available: http://arxiv.org/abs/2105.02540

[17] A.Aimen, S. Sidheekh, V. Madan, and N. C. Krishnan, "Stress Testing of Meta-Learning Approaches for Few-shot Learning".

[18] A.Zeb, F. Din, M. Fayaz, G. Mehmood, and K. Z. Zamli, "A Systematic Literature Review on Robust Swarm Intelligence Algorithms in Search-Based Software Engineering," Complexity, vol. 2023, pp. 1−22, Feb. 2023, doi: 10.1155/2023/4577581.

[19] A.Aggarwal, S. Shaikh, S. Hans, S. Haldar, R. Ananthanarayanan, and D. Saha, "Testing Framework for Black-box AI Models," Feb. 11, 2021, arXiv: arXiv:2102.06166. Accessed: Nov. 15, 2024. [Online]. Available: http://arxiv.org/abs/2102.06166

[20] M. Felderer, E. P. Enoiu, and S. Tahvili, "Artificial Intelligence Techniques in System Testing," in Optimising the Software Development Process with Artificial Intelligence, J. R. Romero, I. Medina-Bulo, and F. Chicano, Eds., in Natural Computing Series. , Singapore: Springer Nature Singapore, 2023, pp. 221–240. doi: 10.1007/978-981-19-9948-2_8.

[21] K. Kalfin, R. A. Ibrahim, and G. S. Laksito, "Optimization of White Box Testing by Utilizing Branching and Repeating Structures in Java Programs Using Base Path," IJMSC, vol. 2, no. 2, pp. 85–89, May 2024, doi: 10.46336/ijmsc.v2i2.98.

[22] A.Althunibat, M. Mahmood, H. Alnuhait, S. Almanasra, and H. A. Al-Khawaja, "Proposed Test Case Generation Model using Fuzzy Logic (TCGMFL)," WSEAS TRANSACTIONS ON COMPUTER RESEARCH, vol. 12, pp. 161–172, Dec. 2023, doi: 10.37394/232018.2024.12.16.

[23] K. Benedict, S. L. Williams, D. J. Smith, M. D. Lindsley, S. R. Lockhart, and M. Toda, "Testing for Blastomycosis, Coccidioidomycosis, and Histoplasmosis at a Major Commercial Laboratory, United States, 2019–2024," Open Forum Infectious Diseases, vol. 11, no. 8, p. ofae448, Jul. 2024, doi: 10.1093/ofid/ofae448.

[24] H. B. Hassan, Q. I. Sarhan, and Á. Beszédes, "Evaluating Python Static Code Analysis Tools Using FAIR Principles," IEEE Access, vol. 12, pp. 173647–173659, 2024, doi: 10.1109/ACCESS.2024.3503493.

[25] T. N. AL-Hadidi and S. O. Hasoon, "Software Defect Prediction Using Extreme Gradient Boosting (XGBoost) with Optimization Hyperparameter," vol. 18, no. 1, 2024.

AUTHORS BIOGRAPHY

[1]Mervat Mohammed Hamed is a master's student in her final year at the University of Mosul, specializing in software engineering. Her research interests include artificial intelligence, software testing, and algorithm optimization. She is particularly focused on leveraging AI techniques to improve software quality and efficiency.

[2]Asil Waleed Ali Al-Naemi is an assistant professor in computer science, specializing in intelligent techniques. Her research interests include image processing, big data, machine learning, and deep learning. She focuses on applying advanced AI methods to solve complex problems in data analysis and image recognition.

Citation of this Article:

Mervat Mohamed Hamid, & A. P. Aseel Waleed Ali. (2025). Impact in Software Testing Using Artificial Intelligence: A Literature Review. *International Research Journal of Innovations in Engineering and Technology - IRJIET*, 9(1), 46-52. Article DOI https://doi.org/10.47001/IRJIET/2025.901006

****

