Proceedings of the 5th International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/112/2025.18116

Artificial Intelligence in Software Testing for Emerging Fields: A Review of Technical Applications and Developments

Yuepeng Ding[1,a,*]

[1]*School of Computer Science and Engineering, Southeast University, Nanjing, China*
a. dingdofta@outlook.com
**corresponding author*

Abstract: The rapid evolution of the Internet industry has transformed software development in emerging fields, making it markedly different from traditional approaches. Consequently, software testing has become increasingly complex. Artificial Intelligence (AI) has introduced innovative technologies and methodologies to software testing, attracting growing research interest in recent years. This study employs a systematic literature review to gather and analyze relevant research on AI applications in software testing over the past four years. Through careful screening, synthesis, and organization of literature, the study provides a detailed analysis of research progress in this field. The review examines 30 carefully selected papers, primarily focusing on the technical advancements and applications of AI in software testing within emerging domains such as cloud technology, autonomous driving, and DevOps. It explores various AI applications in software testing, including test case generation, GUI testing, and defect prediction. As a comprehensive review, this paper summarizes recent progress in AI technology applications for software testing, with particular emphasis on emerging trends. Based on these findings, the study proposes potential research directions and topics, offering valuable insights and laying a foundation for future work in this rapidly evolving field.

Keywords: Artificial Intelligence, Software Testing, Machine Learning, Cloud.

1. Introduction

Software testing, a crucial component of software engineering, can consume up to half of the time and cost in the development process [1]. While continuous integration and continuous delivery/deployment (CI/CD) can enhance testing efficiency and reduce time costs through well-designed automated testing and effective development process management, these approaches still demand ongoing investment from skilled software test engineers.

In recent years, software development has evolved significantly with advancements in cloud native technology, big data, and the Internet of Things (IoT). Simultaneously, new fields like autonomous driving, augmented reality (AR), and virtual reality (VR) have created unprecedented demand for innovative software solutions. These shifts have dramatically altered software products' form, design, development processes, user base, and usage scenarios. Consequently, software testing has become more intricate and multifaceted than ever before. Furthermore, as agile development and DevOps practices gain widespread adoption among development teams, quality assurance has expanded beyond the traditional post-development testing phase. It now encompasses the entire lifecycle of software products—from initial design through development, release, and launch. This expansion necessitates improved test efficiency, broader testing methodologies, and enhanced test scale and coverage to address the increasingly complex challenges of modern software development.

To address the numerous challenges in software testing mentioned earlier, applying artificial intelligence (AI) technology and methods has emerged as a promising solution. AI-based software testing introduces innovative approaches to enhance test efficiency, accuracy, and effectiveness, enabling better management of software testing issues in various new and complex scenarios. With these advancements, developers can construct more comprehensive tests and ensure the reliability of software systems.

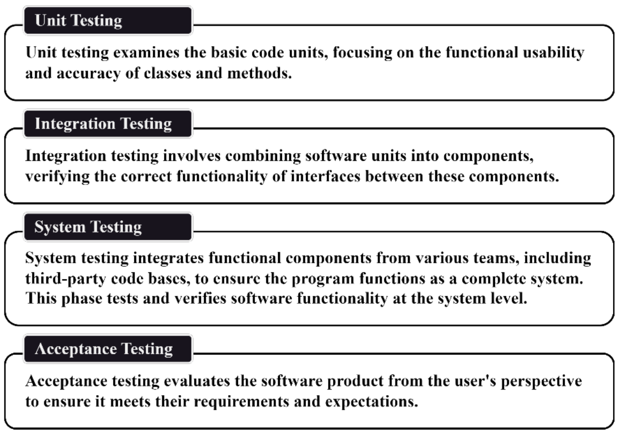
Traditional software testing often relies heavily on manual labor to complete complex tasks, which comes with several drawbacks: lengthy test cycles, inadequate code coverage across all possible branches, and the ever-present risk of human error. While automating some repetitive tasks can be an effective solution, there's still significant room for improvement in overall testing efforts. AI and machine learning-based software testing offers a more sophisticated approach. These technologies can leverage vast amounts of data to automatically generate, construct, execute, and analyze tests. By predicting potential defects based on historical data, AI-driven testing enables faster execution, shorter cycles, improved coverage, and enhanced issue detection. This not only streamlines the testing process but also addresses many of the limitations inherent in traditional manual testing methods.

This article will examine the recent advancements in AI applications for software testing in emerging fields, discussing their benefits, limitations, and future potential through specific case studies. Furthermore, it will categorize the current research focus areas within testing classifications and tasks, evaluate testing effectiveness, identify areas for future coverage, and explore new potential AI applications in software testing for emerging fields.

2. Background and related work

2.1. Overview of software testing

The ANSI/IEEE 1059 standard defines software testing as the process of examining software to identify discrepancies between its current state and its intended state (commonly referred to as defects, errors, bugs, or faults) and evaluating various aspects of the software [2]. As a vital process in the lifecycle of software development, the primary objective of software testing can be concisely summarized as identifying discrepancies, errors, defects, and missing features between the software product and its originally specified requirements.



Unit Testing
Unit testing examines the basic code units, focusing on the functional usability and accuracy of classes and methods.

Integration Testing
Integration testing involves combining software units into components, verifying the correct functionality of interfaces between these components.

System Testing
System testing integrates functional components from various teams, including third-party code bases, to ensure the program functions as a complete system. This phase tests and verifies software functionality at the system level.

Acceptance Testing
Acceptance testing evaluates the software product from the user's perspective to ensure it meets their requirements and expectations.

Figure 1: Testing levels

Software testing can be categorized based on the test task type, testing level, and its association with specific stages in the software development process. It is typically divided into four distinct levels: unit testing, integration testing, system testing, and acceptance testing, as illustrated in Figure 1.

Software testing can also be categorized into two main types: manual testing and automated testing, in which the content of manual testing can be subdivided into many types according to the specific situation. Automated testing, on the other hand, is a programmatic treatment of repetitive work in some manual tests, as shown in Figure 2.

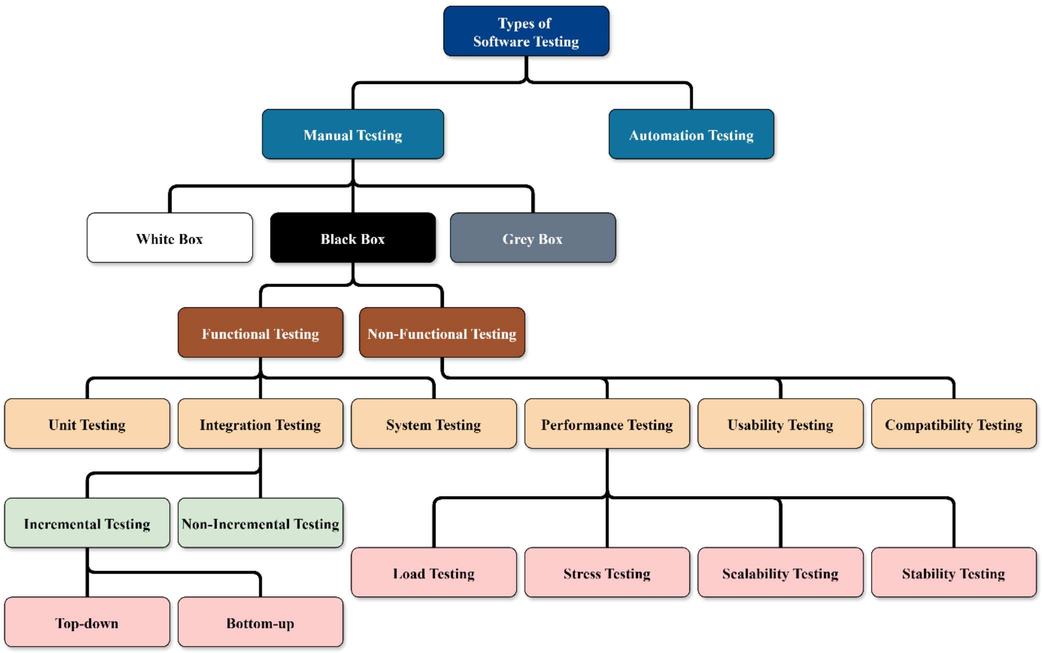


Figure 2: Types of software testing

In the specific classification of testing tasks, white-box testing allows testers to see all the software's code. In contrast, black-box testing conceals the code from testers during the test. This approach prevents testers from designing test cases with preconceived notions, enabling them to identify software issues more effectively. Black-box testing further divides into two parts: functional and non-functional testing. Functional testing verifies the software's functional requirements, while non-functional testing evaluates its performance, usability, stability, and security. For these various categories of testing tasks, automated testing can offer distinct benefits. Likewise, applying AI technology to specific software testing tasks presents multiple opportunities for implementation and improvement.

Additionally, from the perspective of the software testing lifecycle, testing can be divided into distinct phases: requirements analysis, test planning, test case design, test environment setup, test execution, and test cycle closure. Figure 3 illustrates this process.

From these perspectives, AI-driven software testing can enhance the testing process across multiple levels and angles. These enhancements include, but are not limited to, test design, use case generation, test planning, test environment deployment, test execution, defect prediction, and analysis.

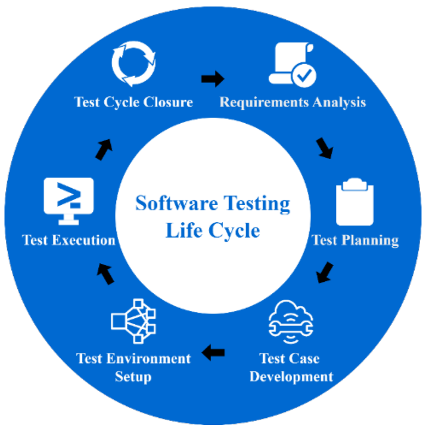


Figure 3: Software testing lifecycle

2.2. Related work

In recent years, significant practical research has advanced the application and development of AI in software testing. Since 2019, IEEE has hosted five international academic conferences focused on AI Testing, exploring topics of "the usage of artificial intelligence for software testing" and "the testing of artificial intelligence methods themselves" [3]. These conferences have yielded 111 relevant papers. There are some examples of specific research literature: Ehab Ebrahim, Mazen Sayed, et al. proposed an AI-based chatbot to improve the organization of release planning and resource management in the process of software development and testing [4]; Faqeer Ur Rehman, Madhusudan Srinivasan, et al. analyzed and studied the application prospects and challenges of transformation testing for machine learning [5]; Yue Ding, Qian Wu, Yinzhu Li, et al. proposed a method based on deep learning to improve the detection of cross-function null pointer risk in unit testing [6]; Yejun He, Muslim Razi, et al. designed a testing framework for autonomous vehicle using AI semantic models [7]; Caglar Osman, Taskin Furkan, et al. developed a software testing platform based on cloud and AI, called ChArIoT [8].

Overall, these research directions primarily focus on specific AI applications in various aspects of software testing. They rarely discuss necessary improvements or adjustments at the algorithmic level for AI in software testing scenarios. Moreover, much of the research is based on actual production and analyzed in conjunction with specific practical scenarios, making these findings more valuable for practical implementation. Additionally, several studies have explored recent advances in applying large deep learning models, including ChatGPT, convolutional neural networks (CNNs) for image recognition and object detection, and generative AI.

Currently, there is no shortage of raw data for testing in real production scenarios, but model training requires processed data. Machine learning training datasets for different scenarios are still insufficient. This part of the work needs to be further developed and improved in the future.

3. Research Methodology

This article employs the SLR (systematic literature review) methodology developed by Angela Carrera-Rivera, William Ochoa, et al. [9].

The review centers on the keyword "Artificial Intelligence in Software Testing," focusing on developments in emerging fields from 2021 to 2024. The study's data sources are confined to five prominent databases: Scopus, Web of Science, EI Compendex, IEEE Digital Library, and ACM Digital Library. Table 1 provides detailed information about these databases.

Table 1: Details of databases

Database	Area	URL
Scopus	Interdisciplinary	http://www.scopus.com/
Web of Science	Interdisciplinary	https://www.webofscience.com/
EI Compendex	Engineering	http://www.engineeringvillage.com/
IEEE Digital Library	Engineering and Technology	http://ieeexplore.ieee.org/
ACM Digital Library	Computing and information technology	https://dl.acm.org/

Based on this topic, the paper presents the following research questions (RQs):

- RQ-1: What are the applications of Artificial Intelligence in software testing?
- RQ-2: What methods of artificial intelligence are commonly used?
- RQ-3: What are the pros and cons of AI testing?
- RQ-4: How to evaluate the effectiveness of AI testing?
- RQ-5: What are the challenges of applying AI methods to software testing in emerging fields?

The search strategy for this article is illustrated in Figure 4. First, the entry and exit criteria for the search are defined and the initial search is carried out within the scope of the selected database. Then, the preliminary results obtained from the search are manually reviewed, and the corresponding Quality Assessment Checklist (QA Checklist) is established to fine screen the selected results. In the end, about 30 articles are obtained for review analysis.

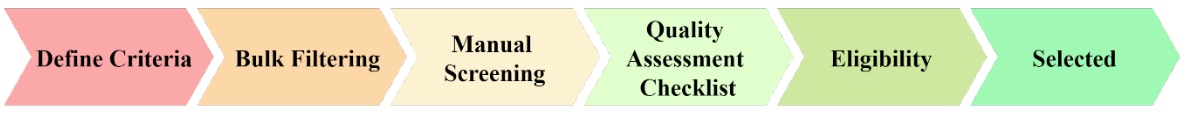


Figure 4: Search strategy

The various criteria used for the search are shown in Table 2.

Table 2: Inclusion and exclusion criteria

Criteria Type	Description	Inclusion & Exclusion
Period	The publication date range of articles	**Inclusion:** From 2021 to 2024
Language	The language of articles	**Inclusion:** In English
Type of Literature	Exclude articles that fall into the gray category	**Inclusion:** Conference Article, Journal Article
Impact Source	Filter for articles that meet the source's impact factor or quartile requirements	**Exclusion:** Impact Score lower than x*
Relevance to RQs	Consider the relevance of articles to RQs	**Inclusion:** Associated with at least 2 RQs

*The x value is adjusted according to the search.

Through the above criteria, a preliminary search was finally carried out in various databases. Although the search strings differ slightly in format and content, it can be ascertained that each database follows the same search criteria. According to the content of the preliminary screening, the retrieved data files are exported from each database. After converting it to a CSV file, it can be used for further fine screening and data processing. The QA Checklist is shown in Table 3. It is used to assess the quality of the obtained literature and to conduct a fine selection based on the examination results.

Table 3: QA Checklist

Check Points	Questions	Level Assessment
Latest Research	Is the study up to date?	1. Before 2022 (Score: 0) 2. 2022-2023 (Score: 1) 3. After 2023 (Score: 2)
Domain Relevance	Whether the study has a strong correlation with software testing, especially in emerging fields?	1. No (Score: 0) 2. Partially (Score: 1) 3. Yes (Score: 2)
Reliability	Are there any issues (shortcomings, limitations) about the validity of the results of the study discussed in the article?	1. No (Score: 0) 2. Partially (Score: 1) 3. Yes (Score: 2)
Clear Topics on Research	Does the article give a specific definition/description of the research purposes/motivations/problems/goals?	1. No (Score: 0) 2. Partially (Score: 1) 3. Yes (Score: 2)
Practicality	Does the article provide specific examples of practice?	1. No (Score: 0) 2. Yes (Score: 2)
Innovation	Whether the article is innovative enough?	1. No (Score: 0) 2. Partially (Score: 1) 3. Yes (Score: 2)

Eventually, 30 articles were finally obtained from the databases as the objects of the review study. In the selected literature, the data were extracted and summarized around the range of indicators mentioned in Table 4. The content of each article is read and analyzed in detail to form relevant report content.

Table 4: Indicators of data extraction

Data Extraction Types	Specifics
General Information	Year of Publication, Publisher, Document Type
AI Methods & Algorithms	Specific AI Approaches
Fields	Related Application Areas
Testing Information	Testing Tasks, Testing Types, Testing Level
Performance	Assessments, Efficiency Improvement, Accuracy

4. Results

4.1. Report

The findings of these studies are summarized as shown in Table 5 through the reading and analysis of the selected research papers.

Table 5: Findings of selected studies

Paper Id	Year	Publisher	Findings
1 [10]	2024	IEEE	The article introduces Mitra, an AI-driven software testing architecture that leverages machine learning and natural language processing techniques. Mitra automatically generates and optimizes test cases, with a focus on edge cases, thereby effectively improving test coverage. Furthermore, the framework incorporates crucial features for predicting potential software defects using historical data.
2 [11]	2024	IEEE	This article details the application of AI and machine learning techniques for optimizing test cases in web applications. The authors introduce a novel model called the Regression Vector Analysis Model (RVA). Implementing this model has enhanced the test sequence quality and improved the test model's performance. The system's accuracy has reached 92%, while the Matthews Correlation Coefficient (MCC) index has achieved 98%.
3 [12]	2024	SPRINGER	The authors present an industry study on AI technologies and tools in GUI testing. Through a survey of 45 respondents, they analyze specific applications of AI in this field. Results reveal that current practitioners' use of AI tools is limited in both scope and depth, indicating a need for further research. The study also highlights AI's broader potential in GUI testing.
4 [13]	2024	IEEE	This paper comprehensively reviews and analyzes the application of LLMs (large language models) in software testing. It examines 102 studies related to software testing using LLMs and discusses representative tasks such as test case design and code repair. Additionally, the paper provides a roadmap and potential exploration pathways for future research.
5 [14]	2024	IEEE	This paper addresses the challenge of deploying AI deep learning models on resource-constrained mobile devices by introducing SafeCompress, a test-driven sparse training framework. SafeCompress automatically compresses large models into smaller ones by simulating attack mechanisms as a security test. The effectiveness and generality of the framework are validated through extensive experiments conducted on five datasets.
6 [15]	2024	MDPI	This article introduces AVESYS, a framework designed to integrate open-source AI algorithms into the testing process for embedded systems. The framework addresses quality assurance challenges in embedded software for autonomous vehicles. The paper evaluates AVESYS's effectiveness in detecting test environment anomalies that could impact results, highlighting AI's significant potential in embedded software testing.
7 [16]	2024	ACM	This paper proposes a defect prediction model based on static code detection, applying machine learning techniques to the process of fault prediction. The study notes that two models have been developed, with results demonstrating high accuracy and practical utility.
8 [17]	2024	IEEE	Test-Driven Development (TDD) is an agile approach that requires developers to design and write functional code based on unit tests. The authors introduce GAI4-TDD, a generative AI tool for test-driven development, which includes a PyCharm plugin. This plugin can generate production code from tests written by TDD developers. The study verifies the practical application of the plugin in three embedded system development environments and provides an initial evaluation of GAI4-TDD's effectiveness.
9 [18]	2024	IEEE	This article introduces AI-driven CI/CD as an innovative approach to managing and updating software projects. Leveraging artificial intelligence, these processes automate the entire software delivery and deployment pipeline. This automation includes parallel testing of multiple versions, resulting in significant time and cost savings.

Table 5: (continued)

10 [19]	2024	IEEE	This article explores the application of generative AI and large language models (LLMs) in unit test generation. The study experimented with using Gen AI to create unit tests for Python code. The authors analyze and discuss the experimental results and outline future work in this area.
11 [20]	2024	IEEE	This article introduces Test-Spark, an open-source IntelliJ IDEA plug-in that uses AI to quickly generate unit tests in an integrated development environment. The authors provide a detailed analysis of its technical implementation principles and compare its advantages to existing tools. Additionally, they discuss future research plans and preliminary results related to the plug-in.
12 [21]	2023	IEEE	This article examines the integration of vision AI and behavior-driven development (BDD) in software test automation. It aims to assess the impact of this integration on test creation, execution, and maintenance. The study investigates and evaluates a Visual Test Framework that incorporates vision AI tools, using a hybrid approach.
13 [22]	2023	IEEE	This paper discusses the specific implementation of machine learning and deep learning technology in software testing and makes a comparative analysis of these aspects.
14 [8]	2023	IEEE	The authors introduce ChArIoT, a cloud-based web platform that uses AI and the MERN stack architecture to mutate Python test code. This scalable and cost-effective solution offers a flexible testing environment, better test coverage, and improved accuracy. By incorporating AI into mutation testing, the platform significantly cuts down on time and resources, boosting the testing process's effectiveness and efficiency.
15 [23]	2023	SPRINGER	This article introduces TestLab, an intelligent automated testing framework. It integrates a range of test methodologies and automates them using AI technology to allow continuous testing of software systems at multiple levels. The framework also includes features such as vulnerability identification and test case generation.
16 [24]	2023	IEEE	The article illustrates and validates the application of AI and machine learning techniques in software defect prediction (SDP) through Nokia's 5G wireless technology test process and extends its concept to the entire software development lifecycle. Finally, the applicability of the method in optimizing the whole test cycle is verified by the results, and its application prospects in other large-scale industrial software development processes are discussed.
17 [25]	2023	SPRINGER	This article describes the development process of EvoMaster, an open-source tool. EvoMaster is designed to automatically generate system-level test cases for enterprise applications.
18 [26]	2023	SPRINGER	This paper studies and discusses the specific algorithms used in software defect prediction, including the War Strategy Optimization (WSO) algorithm and the Kernel Extreme Learning Machine (KELM) for SDP. At the same time, the paper also gives a detailed description and analysis of the application of algorithms in specific Internet of Things (IoT) scenarios.
19 [4]	2023	IEEE	This paper proposes an AI-based chatbot to improve the release planning organization and resource management in the software development and testing process.
20 [6]	2023	IEEE	This paper introduces a deep learning method to enhance the detection of cross-function null pointer risks in unit testing.
21 [7]	2023	IEEE	This article designs and implements a testing framework for autonomous vehicle using AI semantic models and details its design and practice.

Table 5: (continued).

22 [27]	2022	ACM	This article offers an overview of AI's significant impact on the testing process and identifies key AI techniques used in various testing activities. It also provides a comprehensive study of test automation tools, shedding light on the role of AI in industrial testing tools.
23 [28]	2022	IEEE	The authors propose an AI-based approach to behavioral cloning in a specific domain to enhance automated software testing. Using online stores as a case study, they discuss a preliminary investigation of this scheme. The study presents initial results and outlines a roadmap for future research.
24 [29]	2022	IEEE	The article introduces CRISCE, a method for generating accurate car crash simulations from accident sketches. This innovative approach efficiently creates simulations to enhance test cases for critical driving scenarios, particularly collisions. Ultimately, CRISCE aims to improve the testing of autonomous driving systems for vehicles.
25 [30]	2022	IEEE	The article provides a Transformers for GUI testing to solve the problem of automatic test case generation and Flaky Test.
26 [31]	2022	MDPI	This article introduces DePaaS (Defects Prediction as a Service), a cloud-based, global, and unified AI framework for defect prediction. It details the context, use cases, and architectural aspects of DePaaS, along with its specific practical applications.
27 [32]	2022	IEEE	This paper presents specific strategies for improving automatic test case generation algorithms. It proposes a scheme that combines multiple AI approaches to enhance the effectiveness and efficiency of automated test case generation. Additionally, the article offers concrete suggestions and ideas for implementing the algorithm in practice.
28 [33]	2021	SAI	This article examines the application of AI technology in software testing, as of 2021, and explores its potential to streamline the testing process.
29 [34]	2021	ACM	The author proposes a vision for a platform called Test'n'Mo. This platform aims to leverage artificial intelligence and machine learning as the technical foundation for a hybrid system that combines software testing and runtime monitoring. The goal is to enable collaboration between learning agents and runtime monitoring and testing agents to achieve shared testing objectives.
30 [35]	2021	IEEE	This article introduces RiverFuzzRL, an open-source tool that combines reinforcement learning and fuzzing techniques. The authors suggest that this combination could be a significant advancement in software testing. By implementing RiverFuzzRL, a sophisticated AI-guided fuzzing framework, they aim to enhance software testing outcomes. The study explores the method's applicability to various types of software testing and examines potential opportunities and challenges.

4.2. Responses to RQs

4.2.1. RQ-1: What are the applications of Artificial Intelligence in software testing?

For this question, the following aspects can be summarized through the study of the selected literature as the specific application of AI in software testing:

- **Test Case Generation** - This topic has been addressed in several articles and research studies. Machine learning and deep learning techniques enable the use of large datasets as a foundation for test case generation. Furthermore, generative AI shows promise in automating test case creation for more complex scenarios.

- **GUI Test** - This direction allows AI to demonstrate its capabilities more tangibly. As computer vision and object detection technologies continue to advance, machines are becoming increasingly proficient at analyzing and recognizing images. Consequently, GUI testing—a task that once heavily depended on human effort—can now be effectively managed by AI.
- **Defect Prediction** - Several studies have highlighted AI's role in defect prediction. By analyzing vast amounts of historical problem data, AI—particularly through machine learning methods—can significantly aid in predicting software defects. These studies also discuss various defect prediction solutions, such as DePaaS [31], that integrate DevOps and cloud technologies. These integrated approaches offer more comprehensive and reliable defect prediction support through broader and more efficient data collection.

In addition, the studies also cover the use of AI in the following sections: test planning, test prioritization, test case optimization, test execution, mutation test, test coverage, unit test, and embedded test.

4.2.2. RQ-2: What methods of artificial intelligence are commonly used?

For this question, the following specific methods or directions of AI can be summarized from the study of the selected literature: machine learning, deep learning, reinforcement learning, large language models, computer vision, behavioral cloning, generative AI, and natural language processing.

Within the scope of the selected articles, most of the articles focus on exploring the use of AI technology in different aspects of software testing. There is little discussion about specific methods or algorithm improvements in AI, with only a few articles on this:

- Paper [18] discusses AI-driven CI/CD, which mentions the application and improvement of several algorithms, including NLP (Natural Language Processing), CSA (Cuckoo Search Algorithm), ROA (Randomized Optimization Algorithms), and PSO (Particle Swarm Optimization).
- In paper [26], the application of WSO and KELM algorithms for defect prediction on the IoT is discussed.
- A test case generation scheme based on the Behavioral Cloning method is proposed in the paper [28].

4.2.3. RQ-3: What are the pros and cons of AI testing?

The study summarizes some of the advantages and disadvantages of AI testing, as shown in Table 6.

Table 6: The pros and cons of AI Testing

Pros	Cons
- Significant efficiency gains - Labor cost savings - Effective increase in test coverage - Increased depth and breadth of testing - Provide creative and inspiring ideas for complex new scenarios	- Because the objects being tested (i.e., different software products) are very different, there is a certain cost to build an AI model - While there is plenty of test data, there are fewer processed datasets for machine learning or deep learning - The reliability and completeness of AI testing itself has yet to be verified, but the question of how to test AI programs (Test for AI) is actually more complex - The testing tasks that AI technology can currently involve are not comprehensive enough, and there are still many parts that still require human intervention and processing by experienced engineers or experts - AI itself also requires a certain software and hardware resource base, and there is a certain threshold, and the cost of using AI in the actual production environment needs to be considered

4.2.4. RQ-4: How to evaluate the effectiveness of AI testing?

In fact, for applications in the testing field, the validation of the validity of AI tests is no different from the validation of the validity of general testing behaviors, and it is sufficient to verify them using the same or similar methods. For most simple test scenarios or test tasks, the validity verification of AI testing can be demonstrated by a simple comparison of results—that is, the test task is manually completed with traditional testing methods, and if the results are consistent, the test process and conclusions are valid.

For relatively complex scenarios, it is necessary to consider constructing some specific metrics for data statistics and analysis to determine the effectiveness of AI testing. Specific metrics include the following: accuracy, precision, specificity, recall, F1 score, ROC, AUC, ACC, and confusion matrix.

4.2.5. RQ-5: What are the challenges of applying AI methods to software testing in emerging fields?

Emerging areas here include, but are not limited to, the following: DevOps, Cloud-Native, Autonomous Vehicle, IoT, and Games.

In these areas, the requirements and tasks of software testing differ from those of traditional software testing in the past. In more and more scenarios, software testing emphasizes quality assurance throughout the software development lifecycle. Internet companies that are on the front line of testing left and testing right are often mentioned topics. As a result, AI faces some specific challenges in software testing in these emerging areas:

- In the cloud scenario, how to better solve the test problem of distributed systems?
- Sometimes the failure of SDN (software-defined networking) systems is not necessarily a bug in the code at the functional level, but a problem or limitation of hardware resources that causes the failure at the software level. What better solution to this problem is software testing or quality assurance?
- What can AI do in performance testing? What are the advantages of AI technology in the face of system stress testing scenarios with huge traffic volumes?
- Legal and ethical issues still need addressing in the adoption of AI technology for safety and stability testing. This is especially crucial for autonomous driving, as it directly impacts human lives.

5. Analysis and discussion

5.1. General discussion

In the realm of software testing, AI shows great development potential and promising applications. The fresh topics presented annually at IEEE's AI Test conference, along with the steady growth of research interest in various specific directions, demonstrate that this field still has significant room for expansion and innovation.

Through the analysis of research papers in recent years, this paper can summarize some parts of the testing field that are of concern but have not yet been involved in AI testing:

- By leveraging AI for traffic playback, the effectiveness and timeliness of performance testing can be improved and enhanced, which will provide more reliable and rich performance test data.
- By realizing automatic periodic fault injections in large-scale clusters based on machine learning, the concept of chaos engineering can be practiced improving the stability and reliability of the system, and the pain points of self-recovery of large-scale distributed systems can be explored.

- By analyzing and summarizing the LOG information more abstractly based on LLMs, and combining deep learning to summarize past defect cases, the defect location function at the code level is faster and more convenient. This can help developers and testers complete the debug process better and faster.
- By building a more complete automated test cluster based on cloud-native tools such as Kubernetes and Tracing, AI-driven test environment resource management can be implemented to provide more intelligent TaaS (Test as a Service).

Moreover, current research lacks emphasis on enhancing AI algorithms for testing purposes. There remains ample opportunity for further exploration and study in this area, particularly for specific real-world applications.

5.2. AI Testing in different fields

A comparison of research content over time reveals that many scholars and industry professionals have increasingly focused on emerging fields and new concepts in recent years. According to statistics, over 50% of studies relate to content in these emerging areas.

In the realm of DevOps, AI's role in testing primarily centers on process optimization and improvement. Additionally, new approaches may positively impact AI testing efforts in specific scenarios where testing is shifted left or right.

For Cloud Native environments, current research utilizes cloud native and cloud technologies to support AI testing design and implementation. However, there's a notable lack of research on AI testing for cloud services themselves.

Autonomous driving and the Internet of Things have emerged as hot topics in recent years. AI software testing in these fields emphasizes safety and stability. Testing in these areas also focuses on embedded systems tasks. Unlike traditional computer-based tests, tests on small mobile terminals are more performance-oriented. Furthermore, differences in hardware architecture introduce additional challenges to testing.

5.3. Limitations and deficiencies

Despite extensive efforts to search numerous topic-related databases within the given time constraints, some omissions in the search were inevitable. Additionally, space limitations necessitated selecting only a small number of publications closely aligned with the research topic during the fine selection process. In recent years, AI testing has seen abundant research, though many valuable articles were not included in this discussion. Consequently, the inability to analyze a more comprehensive set of reference data represents one of this paper's limitations.

Furthermore, time constraints prevented the verification of experimental validity and authenticity for the articles included in this paper's discussion. Many of these studies provide detailed source code and data for readers to examine and analyze. Therefore, the inability to reproduce and verify some methods represents another limitation of this paper. This aspect of the work can be pursued in the future when conditions permit.

6. Conclusion

Software testing, a critical component of software engineering, plays a pivotal role in ensuring quality throughout the software development lifecycle. As industry evolves, software testing in emerging fields faces increasingly complex and diverse challenges. The application of AI methods to optimize processes and enhance efficiency offers innovative approaches to software testing and quality assurance.

This paper reviews and analyzes research papers from the past four years (2021-2024) on this topic, summarizing the specific distribution and development trends of research content in this field. It also proposes ideas for future research directions and technological development paths.

In contrast to similar review articles, this paper focuses more on the application and development of AI technology in emerging fields. It also analyzes the strengths and weaknesses of specific case studies.

This review offers valuable insights and references for future research. Additionally, the questions raised, and areas identified for further research in the analysis and discussion sections can serve as a foundation for future work.

References

[1] *Myers, G. J., Sandler, C., & Badgett, T. (2011). The art of software testing, John Wiley & Sons.*

[2] *(1993). "IEEE Guide for Software Verification and Validation Plans, ". In IEEE Std 1059-1993 (pp. 1-87). https://doi.org/10.1109/IEEESTD.1994.121430*

[3] *(2019). "Message from Program Chairs, ". In 2019 IEEE International Conference On Artificial Intelligence Testing (AITest) (pp. 11-11). https://doi.org/10.1109/AITest.2019.00006*

[4] *Ebrahim, E. et al. (2023). AI Decision Assistant ChatBot for Software Release Planning and Optimized Resource Allocation. In 2023 IEEE International Conference On Artificial Intelligence Testing (AITest) (pp. 55-60). https://doi.org/10.1109/AITest58265.2023.00018*

[5] *Rehman, F. U., & Srinivasan, M. (2023). Metamorphic Testing For Machine Learning: Applicability, Challenges, and Research Opportunities. In 2023 IEEE International Conference On Artificial Intelligence Testing (AITest) (pp. 34-39). https://doi.org/10.1109/AITest58265.2023.00014*

[6] *Ding, Y., Wu, Q., Li, Y., Wang, D., & Huang, J. (2023). Leveraging Deep Learning Models for Cross-function Null Pointer Risks Detection. In 2023 IEEE International Conference On Artificial Intelligence Testing (AITest) (pp. 107-113). https://doi.org/10.1109/AITest58265.2023.00025*

[7] *He, Y., Razi, M., Gao, J., & Tao, C. (2023). A Framework for Autonomous Vehicle Testing Using Semantic Models. In 2023 IEEE International Conference On Artificial Intelligence Testing (AITest) (pp. 66-73). https://doi.org/10.1109/AITest58265.2023.00020*

[8] *Caglar, O., Taskin, F., Baglum, C., Asik, S., & Yayan, U. (2023). Development of Cloud and Artificial Intelligence based Software Testing Platform (ChArIoT). In 2023 Innovations in Intelligent Systems and Applications Conference (ASYU) (pp. 1-6). https://doi.org/10.1109/ASYU58738.2023.10296551*

[9] *Carrera-Rivera, A., Ochoa, W., Larrinaga, F., & Lasa, G. (2022). How-to conduct a systematic literature review: A quick guide for computer science research. In MethodsX, Volume 9, 2022, 101895, ISSN 2215-0161. https://doi.org/10.1016/j.mex.2022.101895*

[10] *Retzlaff, N. (2024). AI Integrated ST Modern System for Designing Automated Standard Affirmation System. In 2024 4th International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE) (pp. 1386-1391). https://doi.org/10.1109/ICACITE60783.2024.10616416*

[11] *Manojkumar, V., & Mahalakshmi, R. (2024). Test Case Optimization Technique for Web Applications. In 2024 Third International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE) (pp. 1-7). https://doi.org/10.1109/ICDCECE60827.2024.10548325*

[12] *Amalfitano, D., Coppola, R., Distante, D., & Ricca, F. (2024). AI in GUI-Based Software Testing: Insights from a Survey with Industrial Practitioners. In Communications in Computer and Information Science, 2178 CCIS (pp. 328 – 343). https://doi.org/10.1007/978-3-031-70245-7_23*

[13] *Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S., & Wang, Q. (2024). Software Testing With Large Language Models: Survey, Landscape, and Vision. In IEEE Transactions on Software Engineering, vol. 50, no. 4 (pp. 911-936). https://doi.org/10.1109/TSE.2024.3368208*

[14] *Zhu, J., Wang, L., Han, X., Liu, A., & Xie, T. (2024). Safety and Performance, Why Not Both? Bi-Objective Optimized Model Compression Against Heterogeneous Attacks Toward AI Software Deployment. In IEEE Transactions on Software Engineering, vol. 50, no. 3 (pp. 376-390). https://doi.org/10.1109/TSE.2023.3348515*

[15] *Gnacy-Gajdzik, A., & Przystałka, P. (2024). Automating the Analysis of Negative Test Verdicts: A Future-Forward Approach Supported by Augmented Intelligence Algorithms. In Applied Sciences. https://doi.org/10.3390/app14062304*

[16] *Schütz, M., & Plösch, R. (2024). A Practical Failure Prediction Model based on Code Smells and Software Development Metrics. In Proceedings of the 4th European Symposium on Software Engineering (ESSE '23) (pp. 14–22). https://doi.org/10.1145/3651640.3651644*

[17] *Cassieri, P., Romano, S., & Scanniello, G. (2024). Generative Artificial Intelligence for Test-Driven Development: GAI4- TDD. In 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER) (pp. 902-906). https://doi.org/10.1109/SANER60148.2024.00098*

[18] *Mohammed, A. S., Saddi, V. R., Gopal, S. K., Dhanasekaran, S., & Naruka, M. S. (2024). AI-Driven Continuous Integration and Continuous Deployment in Software Engineering. In 2024 2nd International Conference on Disruptive Technologies (ICDT) (pp. 531-536). https://doi.org/10.1109/ICDT61202.2024.10489475*

[19] *Jiri, M., Emese, B., & Medlen, P. (2024). Leveraging Large Language Models for Python Unit Test. In 2024 IEEE International Conference on Artificial Intelligence Testing (AITest) (pp. 95-100). https://doi.org/10.1109/AITest62860.2024.00020*

[20] *Sapozhnikov, A., Olsthoorn, M., Panichella, A., Kovalenko, V., & Derakhshanfar, P. (2024). TestSpark: IntelliJ IDEA's Ultimate Test Generation Companion. In Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion '24) (pp. 30–34). https://doi.org/10.1145/3639478.3640024*

[21] *Ragel, R. K. C., & Balahadia, F. F. (2023). Visual Test Framework: Enhancing Software Test Automation with Visual Artificial Intelligence and Behavioral Driven Development. In 2023 IEEE 15th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM) (pp. 1-5). https://doi.org/10.1109/HNICEM60674.2023.10589222*

[22] *Verma, I., Kumar, D., & Goel, R. (2023). Implementation and Comparison of Artificial Intelligence Techniques in Software Testing. In 2023 6th International Conference on Information Systems and Computer Networks (ISCON) (pp. 1-9). https://doi.org/10.1109/ISCON57294.2023.10112041*

[23] *Dias, T., Batista, A., Maia, E., & Praça, I. (2023). TestLab: An Intelligent Automated Software Testing Framework. In Mehmood, R., et al. Distributed Computing and Artificial Intelligence, Special Sessions I, 20th International Conference. DCAI 2023. Lecture Notes in Networks and Systems, vol 741. Springer, Cham. https://doi.org/10.1007/978-3-031-38318-2_35*

[24] *Stradowski, S., & Madeyski, L. (2023). Can we Knapsack Software Defect Prediction? Nokia 5G Case. In 2023 IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion) (pp. 365-369). https://doi.org/10.1109/ICSE-Companion58688.2023.00104*

[25] *Arcuri, A., Zhang, M., Belhadi, A. et al. (2023). Building an open-source system test generation tool: lessons learned and empirical analyses with EvoMaster. In Software Qual J 31 (pp. 947–990). https://doi.org/10.1007/s11219-023 09620-w*

[26] *Zada, I., Alshammari, A., Mazhar, A.A. et al. (2023). Enhancing IOT based software defect prediction in analytical data management using war strategy optimization and Kernel ELM. In Wireless Netw (2023). https://doi.org/10.1007/s11276 023-03591-3*

[27] *Pham, P., Nguyen, V., & Nguyen, T. (2023). A Review of AI-augmented End-to-End Test Automation Tools. In Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22) (Article 214, pp. 1–4). https://doi.org/10.1145/3551349.3563240*

[28] *Gatt, C., Bugeja, M., & Micallef, M. (2022). Towards Domain-Specific Automated Testing via Behavioural Cloning. In 2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW) (pp. 146-149). https://doi.org/10.1109/ICSTW55395.2022.00037*

[29] *Gambi, A., Nguyen, V., Ahmed, J., & Fraser, G. (2022). Generating Critical Driving Scenarios from Accident Sketches. In 2022 IEEE International Conference On Artificial Intelligence Testing (AITest) (pp. 95-102). https://doi.org/10.1109/AITest55621.2022.00022*

[30] *Khaliq, Z., Farooq, S. U., & Khan, D. A. (2022). Transformers for GUI Testing: A Plausible Solution to Automated Test Case Generation and Flaky Tests. In Computer, vol. 55, no. 3 (pp. 64-73). https://doi.org/10.1109/MC.2021.3136791*

[31] *Pandit, M., Gupta, D., Anand, D., Goyal, N., Aljahdali, H. M., Mansilla, A. O., Kadry, S., & Kumar, A. (2022). Towards Design and Feasibility Analysis of DePaaS: AI Based Global Unified Software Defect Prediction Framework. In Applied Sciences. 2022; 12(1):493. https://doi.org/10.3390/app12010493*

[32] *Olsthoorn, M. (2022). More effective test case generation with multiple tribes of AI. In Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings (ICSE '22) (pp. 286–290). https://doi.org/10.1145/3510454.3517066*

[33] *Minimol, A. J. (2021). Automating and optimizing software testing using artificial intelligence techniques. In International Journal of Advanced Computer Science and Applications, 12(5). https://doi.org/10.14569/IJACSA.2021.0120571*

[34] *Ricca, F., Mascardi, V., & Verri, A. (2021). Test'n'Mo: a collaborative platform for human testers and intelligent monitoring agents. In Proceedings of the 5th ACM International Workshop on Verification and mOnitoring at Runtime EXecution (VORTEX 2021) (pp. 17–21). https://doi.org/10.1145/3464974.3468446*

[35] Paduraru, C., Paduraru, M., & Stefanescu, A. (2021). RiverFuzzRL - an open-source tool to experiment with reinforcement learning for fuzzing. In 2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST) (pp. 430-435). https://doi.org/10.1109/ICST49551.2021.00055