International Journal of Advanced Engineering **ISSN: 2457-0397**

Vol. 4, Issue III, Jul-Sep, 2021 http://bharatpublication.com/ijae

FLEXIBLE METHODS FOR SOFTWARE TESTING USING EMBEDDED ARTIFICIAL INTELLIGENCE IN CHANGING ENVIRONMENTS

Ravindran K

Lecturer in Electronics Engineering

Department of Electronics Engineering, Government Polytechnic College, Palakkad

Kerala 678551, India

ABSTRACT

One useful software testing method is automated testing, which runs test cases automatically using specialized tools. You can write automated testing scripts that simulate user interactions with your program and verify its behavior in place of manually performing tests. Efficiency, accuracy, and test coverage are all significantly increased by this method. The corporate environment is starting to change as a result of automation and the rapid advancement of artificial intelligence. Companies are focusing on using modern AI in combination with automated processes to reach previously unheard-of levels of quality and productivity. This research study examines the transformative potential of AI-driven reporting for test automation. We make it possible for test automation to offer valuable insights in addition to problem identification by utilizing artificial intelligence. Testing embedded software is essential to guaranteeing the performance, functionality, and dependability of real-time systems. Businesses use a variety of software testing techniques to validate complicated embedded system testing scenarios, from automated testing to embedded testing. Explore our all-inclusive solutions for high-quality, error-free releases and manual software testing services. Massive amounts of data are generated daily from a variety of sources, and they must be appropriately monitored, evaluated, reported on, and utilized to inform decisions. Time is becoming an important factor in the deployment of applications that must be extensively tested and meet business requirements as a result of the development of increasingly complex software programs. Because AI can produce faster and more accurate results, it is crucial to software testing. These malfunctions frequently occur during testing and might be dangerous.

Keywords: *AI industry; software applications; test automation; smart manufacturing; machine learning; component behavior, software testing, and artificial intelligence.*

INTRODUCTION

As AI continues to speed up and expand operations across all industries, software testing is also evolving rapidly. Capgemini found that 77% of organizations are investing in AI to improve their quality engineering, showing that AI in software testing is now a real and necessary part of today's world. As you know, teams are under a lot of pressure to release software faster, cover more areas with tests, and find problems earlier in the development process. With AI, they can make repetitive tasks easier, reduce mistakes made by humans, and be more flexible in responding to changing customer needs. Unlike traditional test automation, AI brings a level of 'intelligence' that can learn from past data, adjust to changes in how applications behave, and even predict where future issues might occur. This marks a big change from manual testing to AI-driven testing. AI is already playing a role in many of the apps we use and will soon be an important part of our daily lives and society [1]. According to Oxford, artificial intelligence is: "The theory and creation of computer systems that can do tasks that usually need human intelligence, like speech recognition, seeing things, making decisions, and translating languages." Natural Language Processing, machine learning, deep learning, expert systems, and other ideas are the main parts of artificial intelligence [1, 2]. AI covers a wide range of topics, including smart systems, data analysis, prediction, and decision-making. In recent years, many industries have seen big progress, especially robotics, thanks to machine learning, deep learning, natural language processing, and related algorithms and methods [2, 3]. Machines have outperformed humans in areas like understanding spoken commands, evaluating information, recognizing images, driving, analyzing data, and playing games [3, 4]. The need for advanced materials with great properties is growing, which means more investment is needed for research [3, 4]. The development process is still largely done by skilled scientists in controlled lab settings, a method that hasn't changed much in decades. Even though it is based on clear physical rules and domain knowledge [3, 4], this process is still mostly trial-and-error, which is very time-consuming and hard work [4]. For example, Thomas Edison and his team tested around 6000 materials to find the best catalyst for ammonia synthesis back in the early 1900s [4, 5]. Additionally, too much human input can lead to problems with consistency and unintended bias. Because of these issues, the speed of development is much slower than what manufacturers and customers, who face a complicated and unstable market, need [4, 5]. So, a key goal in this field is to change the current research approach to make material development faster. There has been a lot of interest in experimental automation, which uses advanced tools and statistical methods to automatically choose the best materials [4, 5]. Both academic and business sectors have adopted this, especially in areas like organic chemistry and pharmaceuticals [4, 5]. Automation, which is good at doing repetitive tasks, can greatly increase the number of materials and compounds studied. It also frees researchers from boring tasks, allowing them to focus on more complex and creative problems than before [5, 6]. However, there are still several challenges in making automation more independent [5]. Analyzing large data from microscopy and spectroscopy is slower than the rate at which data is collected [6]. AI allows computers to think and learn. In testing and quality assurance, it helps improve effectiveness and reliability. It can create tests, integrate them into CI/CD pipelines, record ongoing tests, analyze results, and generate reports. Like test automation tools, AI can handle repetitive tasks more efficiently. For instance, instead of writing scripts manually, you can give AI the basic variables, conditions, and acceptance criteria, and it can create test code. AI software testing can do more than just create and run tests. With the right training, it can also review test results, adjust to recent code changes, manage code coverage, and even decide which tests are best for a certain project, module, environment, or test run. Second, relying heavily on specialists to explore vast reaction and chemical spaces reduces efficiency [7, 8]. Based on the previous iteration's findings, a new set of reaction and chemical spaces should be selected for each experiment iteration. Researchers still make this choice in typical automated setups, which might lead to bias and errors [7, 8]. Third, due to the large and high-dimensional nature of the exploration space, it's not feasible for automated robots to list every combination because it would generate too much data, making it hard to find relationships between synthesis, structure, and properties [8]. Therefore, the basis for an Autonomous Experiments Platform (AEP) is the need for intelligent data analysis and decision-making to drive autonomous experiments [7]. In today's fast-paced software development environment, the pursuit of quality assurance excellence never stops [7, 8]. Test automation has become a key part of the software development lifecycle. As companies strive to deliver high-quality software [8]. Automation ensures software dependability by speeding up testing, reducing the need for human labor, and improving fault detection effectiveness [8, 9]. However, test automation's role goes beyond just writing and running scripts. The reporting step, often overlooked and underestimated, is essential in determining the quality of software products [9, 10]. Test reporting has traditionally been a time-consuming process prone to errors and limited in providing useful insights [10]. This study explores how AI is transforming the reporting part of test automation. AI has changed how test results are handled and presented due to its ability to quickly analyze large amounts of data and extract meaningful insights [10]. Intelligent reporting could redefine software testing procedures in the new era brought about by the merging of AI and automation [9]. AI in Automated Testing Artificial Intelligence (AI) has become a powerful force in the ever-evolving fields of software development and quality assurance, transforming the test automation landscape. The combination of AI with automated testing has led to innovative and groundbreaking methods for software testing and validation [8]. Traditional test automation involves creating scripts and test cases that simulate user interactions with a software application. These are run regularly to find issues and ensure software quality. However, AI brings a new level to this process with several benefits [9]:

i. **Augmented Test Script Generation:** AI can automatically create test scripts by looking at the features of an application [10], which saves a lot of time and money that would otherwise be spent on making and keeping test cases.
ii. **Self-repairing Test Automation:** This lowers the work needed to keep tests up to date because AI-powered automation can change test scripts on its own when the program changes [9].
iii. **Enhanced Test Data Management:** AI can create and manage test data without help from humans, which ensures that all parts of the application are properly tested [1-2].
iv. **Intelligent Test Execution:** AI can sort test cases based on how often they are used, how risky they are, or past problems [2,3]. **This helps in testing more thoroughly and finishing tests faster.**

v. **Dynamic Test Reporting:** AI helps in making smart reports and analyzing test results, giving useful insights about how well tests worked, where problems were, and how quality is changing over time [3].

vi. **Predictive Analysis:** AI can spot problems before they happen by looking at old data and finding patterns that might cause issues [3]. This helps in preventing future problems in a proactive way [3].

vii. **Real-time Testing:** AI makes it possible to continuously test and watch the software in real time, making sure its quality stays high throughout the whole development process.

There are several challenges when using AI in test automation, like the need for a lot of training data, ethical concerns, and the risk of biased AI systems [3].

However, when used correctly, AI can greatly improve the effectiveness, accuracy, and speed of test automation [3,4]. This paper shows how AI can be used to predict failures in automotive applications. Testing on-site takes a lot of time and money for car makers, but AI can use existing data to spot and prevent problems [4]. This could speed up the development process and save time. Using advanced diagnostic methods can also help avoid the need for early field testing and the time-consuming creation of diagnostic models. AI predictions can help engineers fix problems before they happen or stop failures from occurring [4,5].

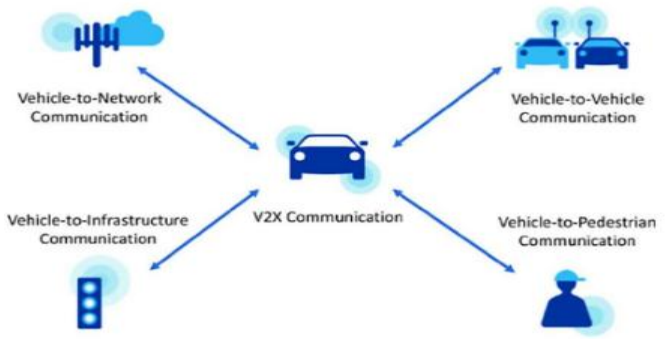
Vehicle-to-Network Communication
Vehicle-to-Vehicle Communication
V2X Communication
Vehicle-to-Infrastructure Communication
Vehicle-to-Pedestrian Communication

Figure 1: AI in Automotive Industry. [1]

CONCEPT OF AUTOMATION

Over the past ten years, automation has become more common as a way to save time and effort. A system that combined human workers with machines has been replaced by a system that uses computers and machines, thanks to automation [7]. In many different areas, automation has made very demanding and repetitive tasks more efficient, and it has also improved the quality of the final results. Automation takes many different forms, but here are some of the most common types:

i. Numerical Control: This includes tools like glass cutting machines, 3D printers, drills, and other devices designed to perform repetitive tasks [5] [6].

ii. Computer-Aided Manufacturing (CAM): Software used in this type of automation includes computer-aided design (CAD), computer-aided design and drawing, and other related tools [6, 7].

iii. Flexible Manufacturing Systems (FMS): This advanced automation system uses robots and other cutting-edge technologies to give consumers more freedom and customization options [7, 8].

iv. Industrial Robots: These robots can be controlled along multiple axes and are used for tasks like welding, assembly, and moving materials [7, 8].

BENEFITS OF AI IN SOFTWARE TESTING

a. Serious cost savings

AI-powered testing can cut costs by doing most boring tasks automatically. Tests are created faster and need less human help, so teams don't need many testers to get the work done. AI also handles a lot of test maintenance, helping to test more areas. This means less need for people, unless it's for big or new tasks. That's one thing you don't have to worry about with your security team. You can keep a small team and still make sure data is safe and private [7, 8].

b. Better software quality

Using AI for testing helps make better software. Its main goal is to improve the software by finding bugs, spotting issues, and checking performance.AI helps create better test cases quickly, which makes testing more complete. It also makes manual testing better by using past data and expert info.AI lets you test more features, run more tests in each sprint, and fix code changes easily. It helps improve the whole testing process and the code itself [7, 8].

c. Improved team collaboration

Thanks to AI, people without technical skills can make and run test cases. This helps everyone on the team understand the product better. Knowing the actual software makes all team members, whether testers or not, aligned with what they're building. At the end of the Software Development Life Cycle (SDLC), you'll see better teamwork, more creative ideas, and a better product [6].

d. Increases test coverage and speed

AI tools look at what the software needs to do, the stories, and the code. This helps create test cases automatically, including tricky edge cases that people might miss.It finds parts of the app that aren't tested well, like rare code paths, and makes tests quickly, even if they're complex. Testers without coding skills can create tests using simple language. They can look at documents or chats to make clear instructions and get complete test cases in seconds. This means more testing done with less effort. Plus, AI works fast, doing tests and creating them much quicker than humans [5].

e. Ability to make data-based decisions

AI can't replace human ideas, but it can automate decisions that take a lot of work. For example, without AI, your team might build tests from scratch. Current tools can't do that, but AI uses machine learning to create structured test cases from simple language. This can help check if an app is easy to use, accessible, and reliable. AI can plan better workflows, detect patterns, and improve tasks based on testing data, user actions, and past experiences. These are things people do, but they don't use all their thinking power. By making these tasks automatic, human testers can focus on making the software better for the business [9].

f. Simplifies and accelerates test maintenance

AI helps with test maintenance by making scripts that can fix themselves when the UI or web page changes. Instead of manually updating test steps or rewriting scripts, AI can decide which tests are most important, find old or duplicate scripts, and adjust tests when new features are added. With smarter ways to find web elements and automatic problem-solving, test suites are much more reliable. This lets QA teams focus on bigger plans instead of constantly fixing tests [7, 8].

TYPES OF AI IN SOFTWARE TESTING

i. **AI-driven test case generation**

AI looks at requirements, user stories, and past defects to create test cases on its own. Using machine learning, it finds high-risk parts of the application and makes detailed tests, including tricky situations that humans might miss. This helps reduce the need for people to create tests and makes testing more complete [5].

ii. **Self-healing test automation**

Regular test automation fails when the UI or web page changes. AI helps by making tests that can change themselves. These tests notice things like layout changes or new text and adjust automatically. This makes testing more reliable and less work to keep up-to-date [4].

iii. **Visual testing with AI**

AI checks how things look on the screen by comparing images and layouts. Instead of checking every pixel, it uses image recognition to spot issues like layout problems, broken designs, or branding mistakes that aren't obvious in the code [7, 8].

iv. **AI-powered test data generation**

Making good test data is hard. AI uses pattern recognition and creates fake data that looks real. It can also make sure test data covers different situations and doesn't include private information [7, 8].

v. **AI-enhanced performance testing**
Testing how fast a system works involves making users act in different ways. AI helps by looking at past performance, predicting where problems might happen, and creating realistic user behavior. It learns from how the system responds to make testing better [6].

vi. **NLP-based test automation**
Natural Language Processing helps testers write tests in plain English. AI reads these instructions, turns them into working tests, and links them with other tools. This makes testing easier for people without technical skills [3-4].

vii. **AI-driven flaky test management**
Sometimes tests fail even when nothing is wrong. AI finds out why tests fail by looking at past runs, logs, and environment details. It suggests fixes or separates failing tests from others. It also learns from mistakes to help prevent them again [5].

viii. **AI in security testing**
AI tools find security issues by looking at code and simulating attacks. Unlike basic security checks, AI learns from past security breaches and can spot new threats. It helps find weaknesses before they can be used by hackers [6].

ix. **AI-assisted test reporting**
Test results can be hard to understand. AI makes reports easier by summarizing key points, showing main risks, and suggesting actions. It can also explain technical results in a way that business leaders can understand [3].

x. **Bias and fairness testing**
AI systems can make unfair decisions. AI in testing helps find biased results by checking models for unfair outcomes. It looks at data, how decisions are made, and the results to make sure everyone is treated fairly [3].

AI Software Testing vs. Manual Testing: Key Differences [7, 8]

Testing Aspect	**Manual Testing**	**AI Software Testing**
Accuracy	Dependent on human judgment; prone to oversight and inconsistencies	AI in software testing detects patterns and defects with higher consistency and precision
Cost Efficiency	More testers needed as scope grows; high overhead for regression tests	AI tools for software testing cut costs by automating repetitive tasks and reducing maintenance
Reliability	Results can vary by tester skill and environment; scripts break easily	Software testing with AI enables self-healing scripts and stable, adaptive pipelines
Test Coverage	Limited by human bandwidth; often misses edge cases	Software testing AI expands coverage with automated test case generation and deeper analysis

Scalability	Difficult to scale with rapid releases; requires added effort each sprint	AI in software test automation scales dynamically, adapting to code changes in real time
Speed	Slower setup and execution; re-testing takes time	Artificial intelligence in software testing accelerates execution and shortens release cycles
Adaptability	Scripts must be manually updated after changes	AI updates regression suites automatically, managing flaky tests with ease

CHALLENGES AND CONSIDERATIONS FOR AI IN SOFTWARE TESTING

1. Not enough quality data

AI models need a lot of high-quality data that is clearly labeled and organized. If this data isn't available or properly managed, the AI tool will make wrong predictions and biased choices. It won't be able to recognize unusual or rare situations [9].

2. Issues around transparency

Advanced AI models, like those used in deep learning, work in a way that's hard to understand. They are often called "black boxes" because it's difficult to see how they make decisions. This lack of clarity can make it hard for testers to know why the AI points out a bug, predicts something, or suggests a certain action. This can make people lose trust in the AI tool and hesitate to use AI-based testing tools [4].

3. Integration bottlenecks

AI tools for software testing may not work well with existing systems and workflows within an organization. This is especially true when trying to fit them into DevOps pipelines, CI/CD processes, or manual testing practices. It often takes a lot of customization to make AI-powered features work with older testing frameworks [5].

4. Skill gaps

Teams using AI for testing need to understand AI, data analysis, and machine learning. Not all QA professionals may have these skills, which can lead to problems in setting up, fine-tuning, and interpreting AI models. Companies may need to provide training to help their teams learn how to use AI and ML in testing. The challenge is bridging the gap between traditional testing skills and AI expertise [4].

5. Significant initial costs

Setting up an AI testing solution can be expensive at first. You'll need money for tools, employee training, and system integration. This can be a problem for smaller teams or companies that might not see returns on their investment quickly enough [6].

6. Adapting to evolving applications

Some AI tools may struggle to work with apps that change quickly and add new features. This is especially true when new behaviors or user scenarios are introduced. Because Agile development often involves frequent changes, your AI testing tools should be able to adapt quickly without needing constant retraining [7].

7. Questions around regulation and compliance

Regulations in certain industries, such as finance, healthcare, and aviation, can limit how much automated testing can be used. These rules, especially around transparency and accountability, can affect the use of AI testing tools. To make the most of AI testing, teams need to set up the tools to meet specific industry regulations [8].

8. Issues with scalability

Not all AI tools can scale as the software being tested becomes more complex. In some cases, the AI's performance might actually get worse as the app grows. If your AI tool was trained on a small codebase, you're likely to face these problems. The tool will have trouble keeping up with the performance and accuracy needed for larger, more complex applications [7, 8].

9. Ethical questions

If the training data is incomplete or biased, the AI can carry those biases into its decisions. This might lead the AI to focus on certain test cases because of past issues in the data. As a result, it could miss important situations and mess up entire testing processes, affecting the fairness and reliability of the results [9].

LITERATURE REVIEW

The literature review is the second part of any research investigation. This section explains whether secondary or primary sources were used as references for the study. It outlines the latest artificial intelligence resources and methods for solving problems in the automotive industry, such as vehicle breakdowns [2-5]. The section is divided into three subheadings. The first introduces AI in the automobile industry. The second looks at previous research on predicting and preventing car breakdowns. The third subheading covers the weaknesses and shortcomings of those earlier studies. This section mainly discusses the pros and cons of using AI to address issues in the automotive sector, especially related to vehicles [2]. One of the key techniques in this part is comparing different algorithms using data from vehicle logs [4].

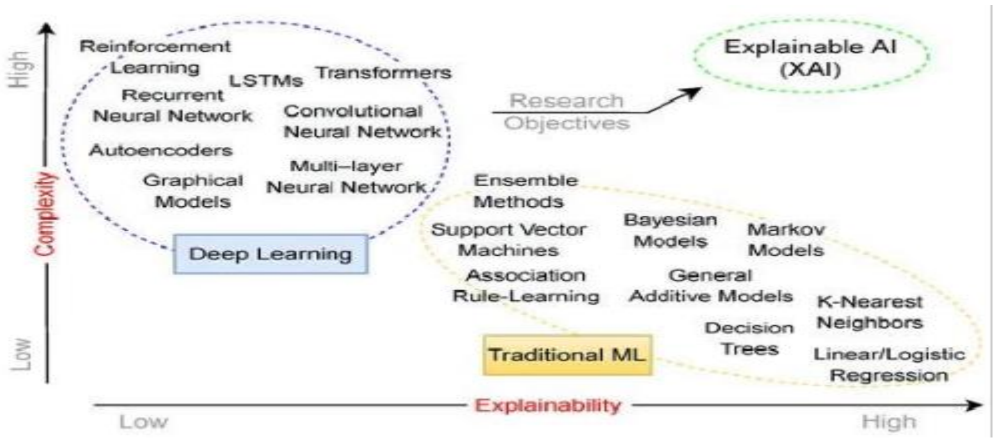

Figure 2: The connection between AI, ML, DL, and XAI.

KEY FLEXIBLE METHODS

1. **Intelligent Test Case Creation and Improvement:** AI looks at requirements, user stories, and past data to make detailed and varied test cases, including tricky edge situations that people might overlook. It can also improve current test sets by finding tests that are not needed, which saves time and resources [7, 8].
2. **Self-Improving Test Scripts:** AI tools can spot changes in the user interface, like a button moving or changing name, and update test scripts automatically. This helps reduce the effort needed to maintain tests, especially in fast-moving agile projects [8].
3. **Predicting Bugs and Setting Priorities:** AI uses past bug reports, code updates, and test results to guess where problems might happen. This helps QA teams focus on the riskiest parts of an app and handle the most important issues first, leading to faster bug finding [5].
4. **Creating Tests with Natural Language:** AI lets non-tech team members, like business analysts, write test steps in simple, easy-to-read language. The AI then changes these into actual test scripts that can be run [8].
5. **Seeing the UI as a User:** AI uses computer vision to check the user interface as a real user would, looking for layout problems, misaligned parts, or font issues across different devices and screen sizes. This goes beyond just checking pixels to ensure a good user experience [7].
6. **Managing Test Data with AI:** AI can create large amounts of realistic and privacy-safe test data, which is important for thorough testing and following rules like GDPR. This takes away the need for manually creating data [7, 8].
7. **Testing Continuously with AI in DevOps:** AI fits into the development process automatically, running relevant tests after each code change and giving real-time feedback to developers. This way, quality is built in throughout the process instead of just checked at the end [8].

8. **Testing with Simulated Hardware:** For complex systems like cars or medical devices, AI helps create and manage simulated models of real hardware. This allows for early and safe software testing without needing physical parts [7, 8].

CONCLUSION

Automated testing for embedded software is a big change for development. It's like moving from fishing in the dark to using a strong net that helps catch more bugs and makes the software work better. Using smart models and algorithms, AI can automatically look at complex data. AI has already shown it can help improve software testing results. In the near future, AI-powered testing will lead the new era of quality assurance work. It will add a lot of value to testing results, control most of the testing areas, and give more accurate results in a fast way. In summary, this study is especially useful for car makers, regular drivers, and especially professional drivers and racing teams. As long as the algorithms are sensitive and specific, AI can be a powerful tool for predicting and preventing failures. The effectiveness of real-time telemetry and data collection depends on how fast the data is analyzed, which humans can't do right now. The car industry could still be affected by new technologies developed outside of it. AI has to keep improving and changing as car technology does. Overall, the future of predicting and preventing car failures in high-pressure situations looks exciting because of the fast growth of AI technology and the big benefits it offers.

REFERENCES

Ahmad, M. F., & Ghapar, W. R. G. W. A. (2019). The era of artificial intelligence in Malaysian higher education: Impact and challenges in tangible mixed-reality learning system toward self-exploration education (SEE). *Procedia Computer Science, 163*, 2–10. **https://doi.org/10.1016/j.procs.2019.12.085**

Benotsmane, R., Kovács, G., & Dudás, L. (2019). Economic, social impacts and operation of smart factories in Industry 4.0 focusing on simulation and artificial intelligence of collaborating robots. *Social Sciences, 8*(5), 143. **https://doi.org/10.3390/socsci8050143**

Bosch, J., Olsson, H. H., & Crnkovic, I. (2018). It takes three to tango: Requirement, outcome/data, and AI driven development. In *SiBW 2018, Software-intensive Business: Start-ups, Ecosystems and Platforms, Espoo, Finland, December 3, 2018* (pp. 177–192). CEUR-WS.org. **http://ceur-ws.org/Vol-2226/paper15.pdf**

Mallozzi, P., Pelliccione, P., Knauss, A., Berger, C., & Mohammadiha, N. (2019). Autonomous vehicles: State of the art, future trends, and challenges. In *Automotive systems and software engineering: State of the art and future trends* (pp. 347–367). Springer. **https://doi.org/10.1007/978-3-030-12157-0_15**

Mascardi, V., Weyns, D., Ricci, A., Earle, C. B., Casals, A., Challenger, M., ... & Winikoff, M. (2019). Engineering multi-agent systems: State of affairs and the road ahead. *ACM SIGSOFT Software Engineering Notes, 44*(1), 18–28. **https://doi.org/10.1145/3317699.3317705**

Poniszewska-Maranda, A., Kaczmarek, D., Kryvinska, N., & Xhafa, F. (2019). Studying usability of AI in the IoT systems/paradigm through embedding NN techniques into mobile smart service system. *Computing, 101*(11), 1661–1685. **https://doi.org/10.1007/s00607-018-0652-x**

Rajan, K., & Saffiotti, A. (2017). Towards a science of integrated AI and robotics. *Artificial Intelligence, 247*, 1–9. **https://doi.org/10.1016/j.artint.2017.03.003**

Vishnukumar, H. J., Butting, B., Müller, C., & Sax, E. (2017, September). Machine learning and deep neural network—Artificial intelligence core for lab and real-world test and validation for ADAS and autonomous vehicles: AI for efficient and quality test and validation. In *2017 Intelligent Systems Conference (IntelliSys)* (pp. 714–721). IEEE. **https://doi.org/10.1109/IntelliSys.2017.8324354**

Watson, M. R., Voloh, B., Thomas, C., Hasan, A., & Womelsdorf, T. (2019). USE: An integrative suite for temporally-precise psychophysical experiments in virtual environments for human, nonhuman, and artificially intelligent agents. *Journal of Neuroscience Methods, 326*, 108374. **https://doi.org/10.1016/j.jneumeth.2019.108374**

Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K., & Zhang, J. (2019). Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE, 107*(8), 1738–1762. **https://doi.org/10.1109/JPROC.2019.2918951**