2025 3rd International Conference on Inventive Computing and Informatics (ICICI) | 979-8-3315-3830-9/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICICI65870.2025.11069964

The Future of Software Testing Automation: Innovations, Challenges, and Emerging Alternatives

Vaidehi Shah[1] (System Engineer)
Independent Researcher[1]
MS In Electrical and Computer Engineering
Wichita State University, Kansas, United States
Email id : shahvaidehi4795@gmail.com

Pranay Yadav [2]*(Assistant Professor)[2]*,
Dept. of Computer Science Engineering (CSE)
Oriental Institute of Science and Technology(OIST),
R.G.P.V. University Bhopal, [M.P.], INDIA
Email:pranaymedc@gmail.com ORCID-0000-0002-9368-8398

***Abstract*—Software testing automation is seeing fast evolution, propelled by innovative developments in artificial intelligence (AI), machine learning (ML), and cloud computing technologies. These advances are transforming the software development environment, providing new opportunities to improve the effectiveness, precision, and adaptability of testing operation. This study investigates the future of software testing automation by analyzing the key advancements that are set to transform testing methodologies in the next years. Prominent advancements include AI-driven test generation methods that utilize machine learning algorithms to automatically produce test cases based on application behavior, as well as self-healing test scripts capable of autonomously identifying and adjusting to alterations in the application interface, thereby substantially minimizing maintenance burdens. Important developments include AI-driven test generation methods that utilize machine learning algorithms to automatically produce test cases based on application behavior, as well as self-healing test scripts that can independently identify and adjust to modifications in the application interface, thereby substantially decreasing maintenance burdens. The incorporation of continuous testing into DevOps pipelines is enhancing the agility and reliability of software delivery, enabling real-time feedback and expedited release cycles. Automation has many benefits, but implementing it is difficult. Maintaining automated test scripts may be complicated and resource-intensive, particularly for rapidly evolving codebases. Due to these constraints, low-code and no-code testing frameworks are becoming popular, democratizing automation by enabling testers without coding skills to build and execute automated tests. Combining these strategies with traditional automated techniques is improving software testing by offering a more complete and effective assessment framework. In research work focus on recent development in the area software testing also compare the different method which is involved in automated software testing.**

Keywords— Software Testing Automation, Artificial Intelligence High implementation costs and DevOps pipelines etc.

I. Introduction

An automated methodology for evaluating software that encompasses the computerization of the verification process to ascertain the software's functionality and compliance with requirements before deployment into production. This enables an organization to do certain software tests more rapidly without the need for human testers, facilitated by automated testing. Automatic testing is more efficacious when used to comprehensive or repetitive test scenarios. [25]. Software testing that is automated makes use of programmed sequences that are carried out by testing tools. Automated testing tools do an analysis of the program, publish the results, and compare the findings with those obtained from prior test runs. After the first creation of an automated test script, it may be used on several occasions. Unit Evaluation, application programming interface (API) testing, and regression testing are only few of the types of automated tests that may be used by an organization to a wide variety of scenarios [28]. The significant advantage of automated software testing is that it reduces the amount of human labor required by converting it into a collection of scripts. Unit testing, for instance, should be examined as a potential candidate for automation if it consumes a significant portion of the resources that are available to a quality assurance (QA) team. Continuous software development approaches include continuous testing, continuous integration (CI), and continuous delivery (CD). Automated tests are an incredibly significant component of these processes since they can be executed repeatedly at any time of the day [30] Fig.1 shows Types of Automated Testing.

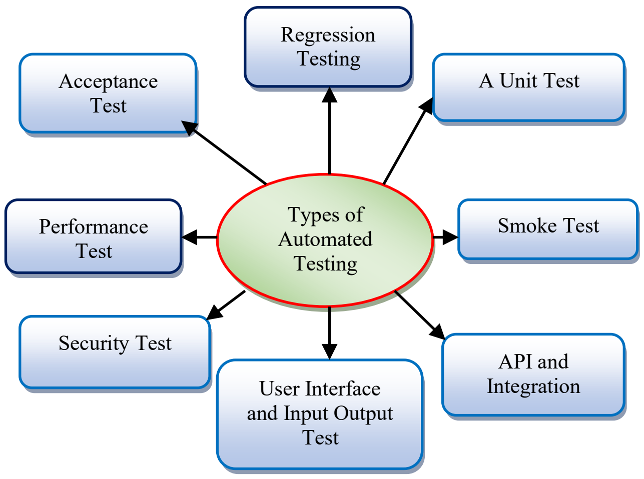

Fig. 1. Types of Automated Testing.

Typically, the automated testing procedure consists of the following steps:

- Choose a testing instrument. The kind of testing being done and if the tool in issue supports the platform the program is being produced on will determine this.
- Describe the automation's scope. This indicates the degree of automation in software testing.
- Create, design, and plan. Creating test scripts and organizing the automation approach are part of this stage.

Automation scripts are used for software testing. Additionally [17], the testing tool must to gather information and provide thorough test results. Upkeep. Newer software builds need automated test scripts to be updated and adjusted as necessary [22].

A The following are the several categories of automated testing:

- Unit Evaluation: Unit testing represents a critical phase in the software testing process, focusing on the assessment of the smallest code segment, known as a unit, that can be logically isolated from the broader codebase [21]. This takes place during the development phase of the application.
- Integration testing : Integration testing represents a phase in the evaluation of software where separate components are combined and evaluated as a whole . This assessment is carried out to evaluate the component's alignment with the specified functional requirements [35].
- Smoke testing: Smoke Analysis is a form of software evaluation that assesses the stability of the developed program. This is the initial evaluation of the program prior to its commercial release.
- Performance Evaluation: Performance testing is a kind of software testing conducted to assess the system's stability and responsiveness under specific load conditions [24].
- Regression testing: RT is a software testing methodology that verifies the functionality of previously built software following modifications, ensuring that the changes have not negatively impacted existing functionalities [23].
- Security Evaluation: safety testing is a kind of software evaluation that detects risks and vulnerabilities inside the application's security architecture [8]. It allows an organization to identify vulnerabilities in the protection architecture and execute corrective actions to rectify the defense inadequacies [10].

A. *UI Evaluation*

UI testing represents a specific category of software testing aimed at verifying that all fields, buttons, and other interface components operate according to their intended design [9].

B. *Automation Approach*

Some of the most common types of automation Approach are:

- Linear framework: This represents the fundamental type of Approach, commonly referred to as the record and playback Approach. Testers develop and implement test scripts for each test case [12]. This solution is primarily appropriate for small teams that possess limited expertise in test automation [14].
- Modular-Based Framework: This Approach categorizes each test scenario into discrete components referred to as modules [16]. Each module operates independently, with distinct scenarios, however all are managed by a singular master script. This methodology necessitates extensive pre-planning and is most appropriate for testers with expertise in test mechanization [18].
- Library Architecture Framework: This framework serves as an enhancement of a modular-based system, demonstrating only slight variations. The task is structured within the test script into functions that correspond with a common goal [20]. The library retains these functions for immediate access when required. This design enhances flexibility and reusability; however, the development of scripts is labor-intensive, making it particularly beneficial for testers who are skilled in automated testing [5].

C. *Software Testing Evolution*

Software testing has seen tremendous change throughout the years. To ensure their code worked properly, developers used to manually inspect it. There was a rise in the need for automated methods to improve and speed up software testing as software became more complicated. Testing is becoming an integral part of modern software development. To guarantee that the software is up to par and meets user needs, a battery of tests is used, including unit tests, integration tests, system tests, and acceptance tests. Better software, produced faster and at a lower cost, is the result of changes to testing procedures.

Organizations must understand the critical significance of test automation in the current scenery. Businesses often recognize the necessity of investing in test automation to maintain high product quality; however, a primary concern is the potential disruption to their release schedule [11].

In the current background, it is essential for companies to continuously innovate and introduce new features ahead of their competitors [13]. This indicates that they lack the capacity to perform all testing procedures with each release. It is essential for not only businesses to select test automation tools that align with their specific requirements, but also for the development of these tools to prioritize a clear return on investment [15].

II. LITERATURE SURVEY

In this section II discuss the different recent research work presented in the last few years in the area of Software Testing Automation.

Alexander Poth et.al (2025) Testing is crucial to software development. Writing and integrating test cases into an automated test suite is laborious. Software must be tested against its needs and specifications. Manually doing it is costly and error-prone. Such work may benefit from automated test-case generating technologies. Deterministic algorithm-driven or non-deterministic AI (e.g., evolutionary and LLM) can generate test cases. When incorporating diverse methodologies into an industrial product test procedure, their strengths and shortcomings must be evaluated. Academic and industrial researchers have produced innovative testing methods and tools, but their efficacy and integration in major industrial settings are still uncertain. This study presents a systematic way to assess and incorporate test-case creation technologies into scalable enterprise setups. Black-box testing of OpenAPI-based REST APIs is the context. To aid IT product and service development. A Volkswagen AG Group IT case study evaluates the Technology Adoption Performance Evaluation (TAPE) technique. Author tested OpenAPI Generator, Evo Master, and Star Coder, which use various technologies. Author's results suggest that these technologies help Volkswagen AG Group IT test engineers specify and develop test cases [01].

Mouna Mothey et.al (2024) Effective software testing is becoming more important as software systems become more relevant and fulfill higher quality criteria. Long-term, reducing resources is preferable than adding test engineers or expanding testing. As software development becomes more complicated and large, automated testing is vital to software quality and dependability. Automated testing frameworks speed up testing, but picking the correct one is tricky. This thesis investigates whether user behavior data may be used to test automation in a data-driven manner. An open-source test automation tool and framework for business intelligence and data warehousing applications is presented here. Data-driven automation can improve test coverage by executing test cases repeatedly, unless there are many. This paper presents a data-driven continuous testing system. This framework runs numerous scripts efficiently. This framework uses Excel to conduct test scenarios. Selenium is used with this framework. Some settings are utilized to run these scenarios. The study shows that the framework can handle many test cases and produce accurate responses. This method removes manual testing in automation [02].

Zubair Khaliq et.al (2022) Because it can run smart factories, operate autonomous cars, make accurate weather predictions, diagnose cancer, and function as personal assistants, artificial intelligence is having a significant influence in a variety of fields, including medicine, the military, industry, domestic life, the legal system, and the arts. In software testing, abnormal activity is looked for and removed. The testing of software is the most challenging and time-consuming of all. By automating some testing procedures, automation technologies increase both the quality of the product and its delivery. When continuous integration and continuous delivery pipelines are introduced, automation systems become less effective. The community of testers is shifting toward artificial intelligence since it can evaluate code for errors and faults more quickly and without the need for human participation. The purpose of this research is to investigate the impact that artificial intelligence technologies have on STLC software testing operations. In addition to this, the paper intends to identify and explain the most significant AI testing issues that software testers face. In addition to this, the paper believes that artificial intelligence will make substantial contributions to the testing of software [03].

III. Testing Process

Test Tool Selection: The selection of the tool will be based on certain parameters. Among the most important factors are: Can we assign competent personnel to automate certain tasks How much money do we have, and is the tool enough for our purposes [19].

Define Scope of Automation: Included in this are several fundamentals, such as the following: the framework should be able to enable automation scripts; there should be less maintenance; Exceptional ROI Test cases that are sophisticated are rare [7].

Planning, Design, and Development: We may do this by using software automation tools like Unit, JUnit, or QUnit, as well as by installing the necessary frameworks or libraries. Then, we can begin to design and construct the test cases [26].

Test Execution: At this point, the test cases will be finally executed. The language used to do this may vary; for example, for.NET, we will use Unit, for Java, JUnit, for JavaScript, Unit or Jasmine, etc. [27].

Maintenance: Reports should be created and recorded following tests so that they may be referenced for future iterations

A. Factors for Choosing an Automation Tool

Some of the factors to consider while choosing the automation program are as follows:

- Ease of use: To build test cases with certain tools, users may need to master a whole new scripting language; to execute those test cases [31], users may need to maintain an expensive and massive test infrastructure; and other tools have a high learning curve [29].
- Support for multiple browsers: Acceptance testing relies heavily on cross-browser testing. It is the user's responsibility to ensure that the tests are easily executable across all supported browsers [32].
- Flexibility: Because no one framework can handle all kinds of testing, it's best to look at what each tool has to offer before making a final decision [33] [34].
- Ease of analysis: Not all instruments have same analytical capabilities. Certain tools provide an appealing dashboard function that displays all test data, including which tests have failed and which have succeeded. Conversely, Some tools may require users to generate and download the test analysis report initially, which can reduce their overall user-friendliness. The selection of a tool depends on the tester, the specific requirements of the project, and the limitations of the budget.
- Cost of tool: Some tools are complimentary while others are commercial; nonetheless, several other aspects must be evaluated prior to determining the appropriateness of using free or paid products. If a tool requires significant time to build test cases for a business-critical process, it is advisable to use a premium solution that can efficiently and rapidly generate test cases.

Availability of support: Free tools primarily provide community assistance, while commercial solutions deliver customer service and training resources such as tutorials and videos. Therefore, it is crucial to consider the intricacy of the testing prior to choosing the suitable gear TABLE I shows difference between manual and automated testing.

TABLE I. DIFFERENCE BETWEEN MANUAL TESTING VS AUTOMATED TESTING

Parameters	Manual Testing	Automated Testing
trustworthiness	Manual testing frequently experiences inaccuracies stemming from human error, which diminishes its reliability.	The execution by external tools and/or scripts enhances its reliability.
Investment	Significant investment in human capital.	Investment on equipment rather than people.
Time efficiency	Manual testing is labor-intensive since test cases are created by human effort.	Automation testing is efficient, since the use of tools accelerates execution relative to manual testing.
Programming knowledge	Programming expertise is unnecessary for composing test cases.	Proficiency in programming is essential for composing test cases.
Regression testing	The first execution of test cases may fail to identify regression flaws owing to the constantly evolving requirements.	Regression testing is conducted to identify defects resulting from modifications in the code.

Effect of security mechanisms on performance

In the first phase of the studies, the bandwidth was configured at 500 Kb/s to simulate a lightly loaded network, indicative of a typical scenario where the network is uncongested. Illustration… …demonstrates the throughput of TCP and UDP traffic types across various security levels. These findings corroborated the overarching patterns indicating that the more robust the security mechanism used, the less effective it became.

IV. CHALLENGES

A. Data Quality

The quality of the data used to train AI systems is crucial to their performance. Unreliable test results and inaccurate forecasts might be the consequence of data inconsistencies, deficiencies, or biases. An example of how an unbalanced dataset might affect the accuracy of testing systems is the possibility of false positives and false negatives produced by such systems [2].

B. Algorithmic Biases

Biased testing algorithms may distort results and make it more likely to ignore faults or make inaccurate evaluations. An AI-powered testing tool may misclassify problems if its predictions are biased by training data Fig. 2 shows Challenges.

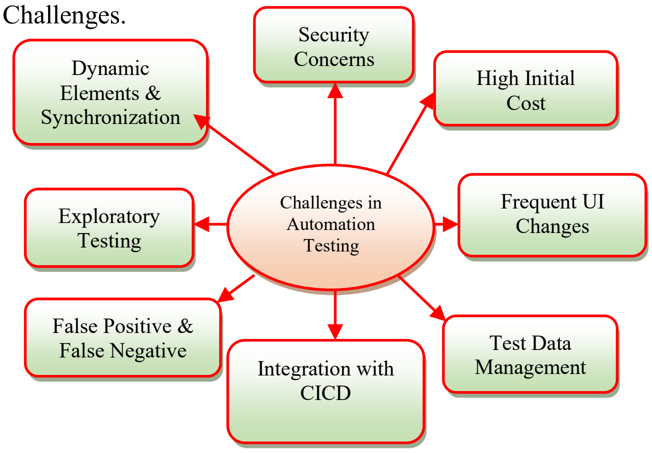

Fig. 2. Types of Automated Testing.

C. Complexity of Tools

The setup and administration of a sophisticated AI-driven testing tool can present significant challenges and require considerable resources. Imagine a company opting to implement a sophisticated AI testing tool. The complicated functionalities and requirements of the tool can pose challenges in terms of configuration, management, and optimization.

D. Integration Challenges

The integration of AI-driven testing into existing testing frameworks and workflows may lead to compatibility issues and disruptions in current procedures due to the inherent complexity involved. To exemplify the issue, compatibility challenges can emerge when attempting to integrate an AI testing solution into an existing Continuous Integration/ Continuous Deployment (CI/CD) pipeline, potentially resulting in delays in the release cycle.

V. EMERGING ALTERNATIVES

The emerging alternatives in testing automation are codeless test , AI and ML based, Mode based Self healing test. There are major type of and next era o software industry automation. In the below Fig. 3 presented the different type and its sub type.

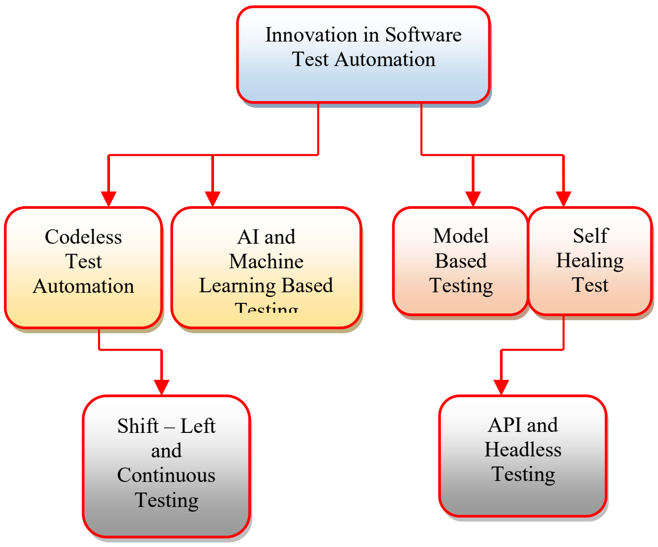
Innovation in Software Test Automation
Codeless Test Automation
AI and Machine Learning Based Testing
Model Based Testing
Self Healing Test
Shift – Left and Continuous Testing
API and Headless Testing

Fig. 3. Innovation in Software Test Automation.

VI. CONCLUSION

The future of software test automation is influenced by rapid technological advancements that tackle persistent difficulties while offering innovative solutions. Traditional automation testing faces challenges such as high maintenance expenses, frequent user interface modifications, and integration complexity; nevertheless, innovative options such as AI-powered testing, codeless automation, and self-healing frameworks are revolutionizing the field. Innovations including model-based testing, robotic process automation, and cloud-based testing are increasing efficiency, scalability, and flexibility. Yet, despite these developments, issues such as ensuring privacy, managing test data, and maintaining dependencies in dynamic contexts remain. Organizations should implement a strategic framework by incorporating contemporary testing methodologies, using AI and machine learning, and emphasizing continuous testing within CI/CD pipelines. The shift-left approach, along with API-first and headless testing, will continue to increase efficiency, facilitating faster and more reliable software delivery. The success of automation in this evolving field will depend on the balance between advanced technology and pragmatic implementation strategy. Fig. 3 shows Innovation in Software Test Automation By adopting innovative solutions and tackling current difficulties, organizations can improve their software testing processes, enhance product quality, and maintain a competitive edge in a rapidly evolving digital environment.

REFERENCES

[1] Poth, Alexander, Olsi Rrjolli, and Andrea Arcuri. "Technology adoption performance evaluation applied to testing industrial REST APIs." Automated Software Engineering 32, no. 1 (2025): 5.

[2] Mothey, Mouna. "Test automation frameworks for data-driven applications." International Journal of Multidisciplinary Innovation and Research Methodology, ISSN: 2960-2068 3, no. 3 (2024): 361-381.

[3] Khaliq, Zubair, Sheikh Umar Farooq, and Dawood Ashraf Khan. "Artificial intelligence in software testing: Impact, problems, challenges and prospect." arXiv preprint arXiv:2201.05371 (2022).

[4] Kim, Ye-Eun, Yea-Sul Kim, and Hwankuk Kim. "Effective feature selection methods to detect IoT DDoS attack in 5G core network." Sensors 22, no. 10 (2022): 3819.

[5] Al-Shareeda, Mahmood A., and Selvakumar Manickam. "Msr-dos: Modular square root-based scheme to resist denial of service (dos) attacks in 5g-enabled vehicular networks." IEEE Access 10 (2022): 120606-120615.

[6] Guşatu, Marian, and Ruxandra F. Olimid. "Improved security solutions for DDoS mitigation in 5G Multi-access Edge Computing." In International Conference on Information Technology and Communications Security, pp. 286-295. Cham: Springer International Publishing, 2021.

[7] Alamri, Hassan A., Vijey Thayananthan, and Javad Yazdani. "Machine Learning for Securing SDN based 5G network." Int. J. Comput. Appl 174, no. 14 (2021): 9-16.

[8] Zhijun, Wu, Xu Qing, Wang Jingjie, Yue Meng, and Liu Liang. "Low-rate DDoS attack detection based on factorization machine in software defined network." IEEE Access 8 (2020): 17404-17418.

[9] Cheng, Jieren, Chen Zhang, Xiangyan Tang, Victor S. Sheng, Zhe Dong, and Junqi Li. "Adaptive DDoS Attack Detection Method Based on Multiple-Kernel Learning." Security and Communication Networks 2018, no. 1 (2018): 5198685.

[10] Hauser, Frederik, Mark Schmidt, and Michael Menth. "Establishing a session database for SDN using 802.1 X and multiple authentication resources." In 2017 IEEE International Conference on Communications (ICC), pp. 1-7. IEEE, 2017.

[11] Nife, Fahad, Zbigniew Kotulski, and Omar Reyad. "New SDN-oriented distributed network security system." Appl. Math. Inf. Sci 12, no. 4 (2018): 673-683.

[12] Benzekki, Kamal, Abdeslam El Fergougui, and Abdelbaki El Belrhiti El Alaoui. "Devolving IEEE 802.1 X authentication capability to data plane in software-defined networking (SDN) architecture." Security and Communication Networks 9, no. 17 (2016): 4369-4377.

[13] Cox, Jacob H., Russell Clark, and Henry Owen. "Leveraging SDN and WebRTC for rogue access point security." IEEE Transactions on Network and Service Management 14, no. 3 (2017): 756-770.

[14] Brief, Solution. "SDN security considerations in the data center." Mike McBride, Editor (2013).

[15] Fathima, Taskeen, and S. Mary Vennila. "Enhancing the Communication of SDN Security with 802.1 X Control Classifier Hilltop (ECSCOCH)." In 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), vol. 1, pp. 1695-1700. IEEE, 2021.

[16] Szymanski, Ted H. "The "cyber security via determinism" paradigm for a quantum safe zero trust deterministic internet of things (IoT)." IEEE Access 10 (2022): 45893-45930.

[17] Kumar, Sonu, Pranay Yadav, and Vijay Bhaskar Semwal. "A Comprehensive Analysis of Lower Extremity Based Gait Cycle Disorders and Muscle Analysis." In Machine Learning, Image Processing, Network Security and Data Sciences: 4th International Conference, MIND January 19–20, 2023, Proceedings, Part I, pp. 325-336.

[18] Tiwari, Sandeep, Nitesh Gupta, and Pranay Yadav. "Diabetes Type2 Patient Detection Using LASSO Based CFFNN Machine Learning Approach."In 2021 8th International Conference on Signal Processing and Integrated Networks (SPIN), pp. 602-608. IEEE, 2021.

[19] Tiwary, Abhigyan, M. Kumar, and Pranay Yadav."Prediction of Covid-19 Patient in United States of America Using Prophet Model." In 2021 International Conference on Advances in Technology, Management & Education (ICATME), pp. 94-99. IEEE, 2021.

[20] Sharma, Bharti, Sachin Kumar, Prayag Tiwari, Pranay Yadav, and Marina I. Nezhurina. "ANN based short-term traffic flow forecasting in undivided two lane highway." Journal of Big Data 5, no. 1 (2018): 1-16.

[21] Yadav, Pranay, Nishant Chaurasia, Kamal Kumar Gola, Vijay Bhasker Semwan, Rakesh Gomasta, and Shivendra Dubey. "A Robust Secure Access Entrance Method Based on Multi Model Biometric Credentials Iris and Finger Print." In Machine Learning, Image Processing, Network Security and Data Sciences: Select Proceedings of 3rd International Conference on MIND 2021, pp. 315-331. Singapore: Springer Nature Singapore, 2023.

[22] Kumar, Alok, Sandeep Kumar Shukla, Archana Sharma, and Pranay Yadav. "A Robust Approach for Image Super-Resolution using Modified Very Deep Convolution Networks."In 2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC), pp. 259-265. IEEE, 2022.

[23] Mishra, Akhil, Ritu Shrivastava, and Pranay Yadav."A Modified Cascaded Feed Froward Neural Network Distributed Denial of Service Attack Detection using Improved Regression based Machine Leaning Approach."In 2022 6th International Conference on Trends in Electronics and Informatics (ICOEI), pp. 1292-1299. IEEE, 2022.

[24] Yadav, Pranay. "Color image noise removal by modified adaptive threshold median filter for RVIN." In 2015 International Conference on Electronic Design, Computer Networks & Automated Verification (EDCAV), pp. 175-180. IEEE, 2015.

[25] Yadav, Pranay, Nishchol Mishra, and Sanjeev Sharma. "Depth Analysis on Recognition and Prevention of DDoS Attacks via Machine Learning and Deep Learning Strategies in SDN." In 2024 2nd International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS), pp. 612-619. IEEE, 2024.

[26] Kumar, Sonu, Pranay Yadav, and Vijay Bhaskar Semwal. "A Comprehensive Analysis of Lower Extremity Based Gait Cycle Disorders and Muscle Analysis." In Machine Learning, Image Processing, Network Security and Data Sciences: 4th International Conference, MIND January 19–20, 2023, Proceedings, Part I, pp. 325-336. Cham: Springer Nature Switzerland.

[27] Chandu, H. S. "A Survey of Memory Controller Architectures: Design Trends and Performance Tradeoffs." Int. J. Res. Anal. Rev 9, no. 4 (2022): 930-936.

[28] Chandu, Hiranmaye Sarpana. "Advancements in Asynchronous FIFO Design: A Review of Recent Innovations and Trends." International Journal of Research and Analytical Reviews 10, no. 2 (2023): 573-580.

[29] Chandu, H. S. "A Survey of Semiconductor Wafer Fabrication Technologies: Advances and Future Trends." Int. J. Res. Anal. Rev 10, no. 04 (2023): 344-349.

[30] Chandu, Hiranmaye Sarpana. "Enhancing Manufacturing Efficiency: Predictive Maintenance Models Utilizing IoT Sensor Data." International Journal of Scientific Research and Technology (IJSART) 10, no. 9 (2024).

[31] Chandu, Hiranmaye Sarpana. "Efficient Machine Learning Approaches for Energy Optimization in Smart Grid Systems."

[32] Chandu, Hiranmaye Sarpana. "Performance Evaluation of AMBA-3 AHB-Lite Protocol Verification: Techniques and Insights." International Journal of Advanced Research in Science, Communication and Technology (2024).

[33] Chandu, Hiranmaye Sarpana. "A Review of IoT-Based Home Security Solutions: Focusing on Arduino Applications." TIJER–Int. Res. J 11, no. 10: a391-a396.

[34] Chandu, H. S. "A Review on CNC Machine Tool Materials and Their Impact on Machining Performance." International Journal of Current Engineering and Technology 14, no. 5 (2024): 313-319.

[35] Chandu, Hiranmaye Sarpana. "Advanced Methods for Verifying Memory Controllers in Modern Computing Systems." International Journal of Advanced Research in Science, Communication and Technology 4, no. 2 (2024): 377-388.

[36] Maurya, Sweta, Shilpi Sharma, and Pranay Yadav."Internet of things based air pollution penetrating system using GSM and GPRS." In 2018 International Conference on Advanced Computation and Telecommunication (ICACAT), pp. 1-5. IEEE, 2018.