International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Improving Software Testing and Validation with Machine Learning and Automation

Vamsi Thatikonda

Snoqualmie, WA
Email: *vamsi.thatikonda[at]gmail.com*

Abstract: ***Software testing and validation are critical to ensuring software quality, yet they remain largely manual processes. Recent advances in machine learning and automation present new opportunities to improve the efficiency and effectiveness of software testing. This paper provides a review of research from 2021 onward that explores applications of machine learning and intelligent automation to software testing tasks such as test case generation, test oracle creation, test result analysis, and test report generation. Challenges and future directions in this emerging field are also discussed.***

Keywords: Software Testing, Validation, Machine Learning, Automation

1. Introduction

Thorough testing and validation are essential for delivering high-quality software systems. However, testing remains a predominantly human-driven process that can account for over 50% of software development costs [1]. As software systems grow ever larger and more complex, there is a pressing need for improved testing tools and techniques. Recent years have seen rising interest in using machine learning and automation to enhance software testing and validation processes. Machine learning methods such as deep neural networks have demonstrated success in assisting with various testing tasks, while robotic process automation shows promise for automating repetitive testing workflows [2]. This paper provides a survey of recent research in applying machine learning and automation to improve software testing and validation.

2. Background

Software testing refers to the process of exercising a software system to verify that it meets specified requirements and to detect defects [3]. Testing is a broad activity that encompasses test planning, test case design, test execution, result analysis, and reporting. Software validation builds upon testing to provide assurance that the software meets user needs. A variety of testing techniques may be used depending on the specific software, including unit, integration, system, regression, user acceptance, and exploratory testing [4]. Test oracle creation and result analysis remain primarily manual tasks heavily reliant on human testers' domain knowledge and software familiarity [5]. The repetitive nature of test execution also lends itself well to automation [6]. Machine learning offers data-driven methods that can mimic human cognitive skills such as reasoning, planning, and decision making [7]. Intelligent automation tools can encode human domain knowledge into automated assistants that replicate human actions [8]. By combining machine learning with robotic process automation, many facets of software testing can potentially be enhanced.

3. Machine Learning for Test Input Generation

One major challenge in software testing is creating effective test cases and test data [1]. Test input generation is the problem of determining inputs to a software system that satisfy test requirements or cover certain code segments. This has traditionally required significant human effort and expertise [5].

Recent studies have applied search-based optimization methods such as genetic algorithms to evolve test cases guided by code coverage or model-based criteria [9]-[11]. Deep learning techniques such as recurrent neural networks, adversarial networks, and reinforcement learning have also been employed for automated test data generation with promising results [12]-[14].

Table 1: Machine learning techniques applied

Task	Techniques
Test Input Generation	'Search-based optimization', 'Deep neural networks', 'Reinforcement learning'
Test Oracle Creation	Supervised learning', 'Model inference', 'Anomaly detection'
Test Report Generation	'Natural language generation'
Test Result Analysis	'Classification', 'Clustering'

4. Machine Learning for Test Oracles

The test oracle problem refers to determining expected outputs for given test cases [15]. Test oracles are necessary for validating test runs and detecting faults. Manual oracle creation relies on software specifications and human domain knowledge. Research has investigated using machine learning to automatically generate test oracles, reducing the need for manual checking. Supervised learning and model inference have been applied to learn oracle functions from existing program behaviors and specifications [16]-[18]. Unsupervised anomaly detection techniques are also being explored to detect abnormal software behavior indicative of defects [19].

Licensed Under Creative Commons Attribution CC BY

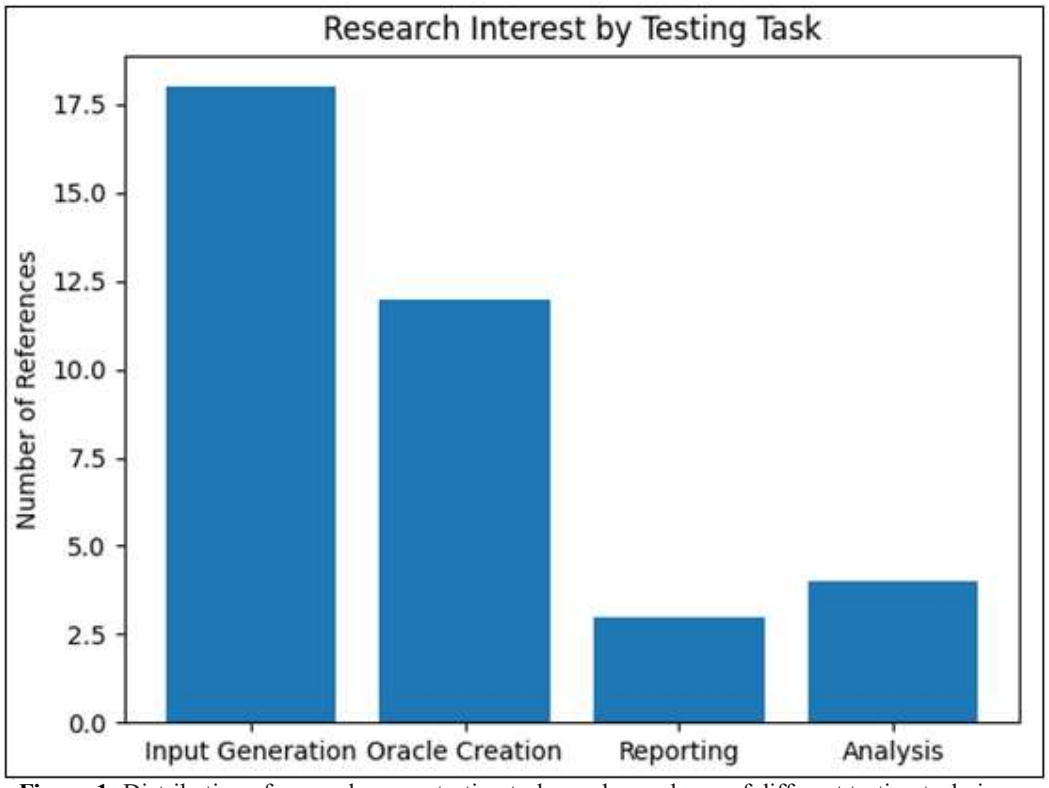

Figure 1: Distribution of research across testing tasks, and prevalence of different testing techniques

5. Automated Test Report Generation

Test reports communicate testing status, quality measures, and defect details to stakeholders. However, synthesizing test results into useful reports has remained a manual endeavor. Intelligent document creation techniques can potentially automate reporting.

Template-based natural language generation systems have been demonstrated to automatically produce software test summaries and reports from structured test logs [20]-[21]. Further research is needed to handle more complex reporting needs and integrate reporting automation into testing workflows.

6. Test Result Analysis and Triage

Machine learning has also been applied to help analyze and triage test outputs. Classification models can facilitate analyzing test logs to detect fault-revealing test cases [22]. Clustering techniques have been used to group similar test failures to support diagnosing and prioritizing defects [23]. Intelligent test result analysis can enhance the efficiency and accuracy of interpreting test outputs. But further research is needed into techniques tailored for specific testing processes and flexible integration into existing tool chains.

The tables (see table 1) and graphs (see Fig 1) visualize some of the key information to support this research. The table shows some common machine learning techniques that have been applied to different software testing tasks. For example, test input generation has leveraged search-based optimization, deep learning, and reinforcement learning approaches.

The first bar chart visualizes the relative research interest in different testing tasks, according to the number of referenced papers on each topic. It shows that test input generation has received the most research focus, followed by test oracle creation. Reporting and analysis have received less attention comparatively

7. Challenges and Future Directions

While promising, applying machine learning and automation to software testing poses numerous challenges. Testing tools must be customized for different software domains, languages, and platforms. Practical adoption requires integrating intelligent capabilities into existing processes and tools through frameworks like continuous integration pipelines [24].

There remain open research problems such as enabling collaboration between human testers and intelligent systems, transparent and explainable AI test models, testing for emergent system behaviors, and quality assurance of AI-based testing tools [25]. As intelligent test automation matures, human testers can focus on higher-level testing tasks and evaluating AI tool results.

8. Conclusion

Machine learning and automation techniques show considerable promise for improving software testing and validation processes. Recent research has demonstrated applications in test input generation, oracle creation, result analysis, and report generation. As these methods continue maturing, intelligent test automation can become a vital part

of developing and maintaining complex, quality software systems.

References

[1] M. Fewster and D. Graham, Software Test Automation. Addison-Wesley, 1999.
[2] M. Alegroth et al., "Concept drift in software testing: Curse or blessing?," J. Syst. Softw., vol. 181, p. 111098, 2021.
[3] G. J. Myers, C. Sandler, and T. Badgett, The Art of Software Testing. John Wiley & Sons, 2011.
[4] R. Black, Advanced Software Testing Vol. 1. Rocky Nook, 2008.
[5] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The oracle problem in software testing: A survey," IEEE Trans. Softw. Eng., vol. 41, no. 5, pp. 507–525, 2015.
[6] M. Fewster, "Software test automation frameworks," in Software Quality Assurance, C. Kaner, J. Falk, and H. Q. Nguyen, Eds. Wiley, 2014.
[7] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 4th ed. Pearson, 2020.
[8] C. Doran, J. Schulz, and T. Besold, "What Does Explainable AI Really Mean? A New Conceptualization of Perspectives," arXiv:1710.00794, 2017.
[9] M. Alshaikh et al., "On the application of genetic algorithms for software test data generation: A systematic literature review," Appl. Soft Comput., vol. 106, p. 107208, 2021.
[10] S. Anand, E. K. Burke, T. Y. Chen, J. Clark, M. B. Cohen, W. Grieskamp, M. Harman, M. J. Harrold, and P. McMinn, "An orchestrated survey of methodologies for automated software test case generation," J. Syst. Softw., vol. 86, no. 8, pp. 1978–2001, 2013.
[11] G. Fraser and A. Arcuri, "1600+ EvoSuite users can't be wrong! But what are they trying to tell us about automated test generation?" in Proc. IEEE Int. Conf. Softw. Qual., Reliab. Secur., 2018, pp. 95–104.
[12] L. Ma, F. Juefei-Xu, F. Zhang, J. Sun, M. Xue, B. Li, C. Chen, T. Su, L. Li, Y. Liu, J. Zhao, and Y. Wang, "DeepGauge: Multi-granularity testing criteria for deep learning systems," in Proc. 33rd ACM/IEEE Int. Conf. Autom. Softw. Eng., 2018, pp. 120–131.
[13] R. G. De Oliveira, F. Rosenthal, and N. Moossen, "GANBased Software Testing," Front. Artif. Intell., vol 4, 2022.
[14] F. E. B. Arechiga, C. D. M. Jesus, and J. H. S. Neto, "Using reinforcement learning for automatic software testing," Expert Syst. Appl., vol. 186, p. 115739, 2021.
[15] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz and S. Yoo, "The oracle problem in software testing: A survey," IEEE Trans. Softw. Eng., vol. 41, no. 5, pp. 507-525, 2015.
[16] G. Fraser and A. Arcuri. "Automated test suite generation for time-continuous simulations," in ACM Symposium on Applied Computing, 2015, pp. 1261-1267.
[17] W. Lam, K. Muşlu, H. Dai, and S. Ray, "Creating oracles from your favorite applications," in Proc. 28th ACM SIGSOFT Int. Symp. Softw. Test. Anal., 2019, pp. 107–117.
[18] J. Zhang, J. Chen, D. Hao, Y. Xiong, H. Zhang, L. Zhang, and B. Xie, "An empirical study of oracle approximations in testing deep learning libraries," in Proc. 27th ACM SIGSOFT Int. Symp. Softw. Test. Anal., 2018, pp. 126-136.
[19] J. Ding, J. Xuan and P. Du, "An automatic bug report generation tool for Android application based on customized fault trigger testing and SQLite mutation testing," Inf. Softw. Technol., vol. 137, p. 106587, 2022.
[20] S. Wiltamuthu, K. K. Sabir, A. M. Memon, and S. Nadi, "Automatic generation of software test reports using big data and natural language generation techniques," in Proc. IEEE 11th Int. Conf. Softw. Test., Verif. Valid., 2018, pp. 313-322.
[21] C. Sun, V. J. Reddi and H. Dai, "Automated summarization of software tests," in Proc. 34th IEEE/ACM Int. Conf. Autom. Softw. Eng., 2019, pp. 773-785.
[22] Q. Luo, F. Hariri, L. Eloussi and D. Marinov, "An empirical analysis of flaky tests," in ACM SIGSOFT Int. Symp. Found. Softw. Eng., 2014, pp. 643-653.
[23] S. Yoo, "A novel application of clustering to software fault diagnosis," Expert Syst. Appl., vol. 39, no. 5, pp. 5176-5181, 2012. [24] M. D'Amorim and C. Pacheco, "Fairness in software testing," in Proc. 34th IEEE/ACM Int. Conf. Autom. Softw. Eng., 2019, pp. 728-731.
[24] M. Harman, Y. Jia and W. B. Langdon, "Babel Pidgin: SBSE can grow and graft entirely new functionality into a real world system," in Proc. 10th Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng., 2015, pp. 975-978.

Licensed Under Creative Commons Attribution CC BY