IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Advancing Web Application Performance Testing: A Review of Machine Learning, Deep Learning, and Reinforcement Learning Approaches

Adam Hamadi (Qu ID: AH1705119)[1]
[1]Qatar University, Doha, Qatar (e-mail: ah1705119@qu.edu.qa)

ABSTRACT Advancement in artificial intelligence (AI) have opened many new opportunities to enhance web application performance testing. The literature review investigates the applications of Machine Learning, Deep Learning, and Reinforcement Learning techniques in both automating and improving web performance evaluation. This review shows how ML models allow for early prediction of performance issues using structured design metrics, while DL methods autonomously generate complex test cases to discover worst case performance scenarios without explicit system modeling. RL approaches are shown to facilitate dynamic exploration of web systems through intelligent workload generation and black box settings. A comparative analysis is performed to showcase their strengths, limitations, and best use cases of each AI technique. Furthermore, this review identifies the challenges that these AI techniques face, including data scarcity, model generalization, computational overhead, and lack of standardized benchmarks, offering directions for future research. This study demonstrates that AI techniques, when appropriately leveraged, can significantly advance the scalability, automation, and effectiveness of web application performance testing.

INDEX TERMS Web Application Performance Testing, Machine Learning, Deep Learning, Reinforcement Learning, Automated Testing, Intelligent Test Generation, AI in Software Testing, Performance Bottleneck Detection

I. INTRODUCTION

A. BACKGROUND

In recent years, web applications have witnessed a rise an independent development that led to a widespread adoption of distributed systems that allow for continuous integration and service delivery [1]. Web technologies have rapidly advanced, leading to more capable web applications like desktop software, while maintaining the simplicity of deployment and access [2]. Nonetheless, the diversity in the technologies used across the client and server sides modern web systems makes performance testing an undeniable challenge [2]. As performance testing gets to remain critical, multifactor specifically in enterprise applications where poor responsiveness potentially led to disruptions in key business functions [3]. As such, considering high performance has become a pivot for the reliability and success of today's web-based systems.

B. RELATED WORK

Despite the availability of automated frameworks like selenium [4], Performance testing of web application still relies heavily on manual testing. Testers are required to simulate user actions through the use of scripts, which makes it both time consuming and dependent on the main expertise [5][6]. In addition, the significant rise in web application technologies forces a revision of the already existing scripts for every new version [7]. The use of alternative strategies, such as random based testing [8] [9], attempts to create input sequences automatically. However, this usually leads to invalid or unrealistic operations (e.g., clicking input fields), with invalid exploration that often miss the deeply nested or rarely visited web pages. Model based testing approaches [10][11][12] try to guide that's generation using navigation models, but these models only cover a subset of the application, struggle with content that is either dynamic or complex business logic that needs long and specific action sequence [7] [13]. Given these

limitations, there is an undeniable need for intelligent, data-driven techniques that are capable of autonomously generating valid, effective, and high coverage test cases. Machine learning, Deep Learning, and Reinforcement Learning offer promising results, and then hands capabilities that allow for higher adaptation the complex application behaviors and a reduction of human intervention and performance testing.

C. OBJECTIVE AND CONTRIBUTION

The aim of this review paper is to investigate and compare the use of different Artificial Intelligence techniques, focusing on Machine Learning (ML), Deep Learning (DL), and Reinforcement Learning (RL) in the context of web application performance testing. This is done by analyzing six recent studies that explore the use of AI-Driven approaches to address the limitations of traditional and automated testing strategies, specifically in terms of adaptability, coverage, and efficiency. While previous surveys focus on investigating one single technique or general software. This work will investigate across the three different AI paradigms. This review will provide a review of the strengths, limitations, and practical considerations of ML, DL, and RL, while also performing a comparative analysis. The objective is to guide researchers and practitioners to select the most appropriate AI-based approach built-on testing goals, system complexity, and resource constraints.

D. ORGANIZATION

The remainder of the paper is constructed in the following way: Section 2 presents an overview of web application performance testing and the role of intelligent testing techniques. Section 3 provides a literature review of the recent studies that utilize ML, DL, and RL. Section 4 presents a comparative analysis of the studies. Section 5 discusses the challenges and research gaps. Lastly, Section 6 conclusion of the paper.

II. WEB APPLICATION PERFORMANCE TESTING

A. BACKGROUND

Performance, which is also referred to as efficiency in software quality attribute classification [14], highlights the effectiveness of what the system accomplishes in terms of functionality. It highlights time- and resource-bound aspects of a system's behavior. Performance is commonly measured through matrices such as throughput, response time, and resource utilization. In terms of web applications, performance is incredibly essential, as inefficient, slow, or unreliable systems can negatively impact the user's experience, engagement, and business outcomes. Performance analysis usually focuses on the following: (i) examine and measure these performance matrices, (ii) track and detect functional issues that might occur under specific execution conditions, such as high user load, and (iii) recognize violations of performance requirements [15].

B. AI-DRIVEN TESTING

Artificial intelligence (AI) techniques have gained increasing popularity when it comes to exploring the effectiveness and efficiency of web application performance testing. Among the many branches of AI, Machine Learning (ML), Deep Learning (DL), and Reinforcement Learning (RL) have shown significant promise when it comes to addressing the limitations of traditional testing approaches. [16] [17] [3] ML techniques involve algorithms that are capable of learning patterns from historical data and making predictions or decisions without the explicit need for programming [16] [18]. The context of performance testing, ML models can predict bottlenecks, classify performers anomalies, and predict system behavior under various load conditions based on observed metrics such as response time and throughput. DL is a specialized subset of ML, it utilizes artificial neural networks with multiple layers to model complex relationships within large datasets [17][19]. DL models, specifically recurrent and convolutional architectures, are highly capable when it comes to capturing intricate temporal or spatial dependencies in web performance data. RL differs from ML and DL; it focuses on learning optimal behaviors through interactions with an environment. In the context of performance testing, RL agents can autonomously explore web applications, generate efficient test cases, and adapt testing strategies in real time by maximizing reward signals such as achieving high coverage or discovering performance degradations. [3] [2]

III. LITERATURE REVIEW

A. ML-BASED APPROACHES

1) Paper 1: A Novel Approach for Evaluating Web Page Performance Based on Machine Learning Algorithms and Optimization Algorithms [20]

This study was conducted in the year 2025 with the purpose of creating a machine learning centric framework for predicting web page performance, while also aiming to support the early stages in modern web development. Instead of relying on conventional runtime testing, the authors utilize the classical machine learning algorithms trained on a diverse set of web metrics in order to classify web pages according to performance quality. The system is built on a pipeline that includes collection, preprocessing, feature selection, and algorithmic evaluation. Figure 1 below shows the complete ML workflow. The dataset (contains 59 web performance attributes such as content relevance, technical optimization) is preprocessed, encoded, and split into training and testing subsets. Feature selection techniques are also applied to reduce the dimensionality and improve generalization. Finally various machine learning algorithms are trained and tested to predict webpage performance. The authors of this paper conducted multiple classical algorithms evaluation, including techniques such as support vector machine (SVM), random forest (RF), logistic regression (LR), Naïve Bayes (NB), and K-Nearest Neighbors (KNN). After the feature selection step, the SVM model demonstrated the heighest predictive accuracy of 89% with improvements observed across all

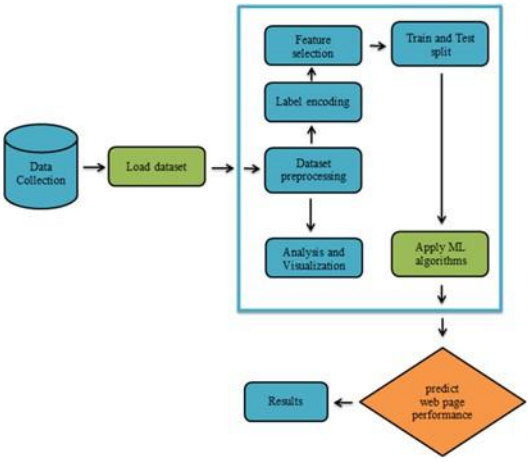

FIGURE 1. ML-based performance prediction approach [20]

evaluation metrics. The significance of these results was confirmed using the Friedman and Wilcoxon Signed-RANK TESTS, confirming that the SVM outperformed other models with confidence.

TABLE 1. Classification performance of evaluated models

Model	Accuracy (%)	Precision (%)	Recall (%)
SVM	89	88	89
Random Forest	81	80	81
Logistic Regression	77	76	77
k-Nearest Neighbors	74	73	74
Naive Bayes	71	70	71

The study demonstrated how valuable classical machine learning algorithms are, not only for retrospective analysis, but as a primary instrument for proactive performance prediction. By modeling performance analysis based on early design metrics, this approach enables developers to optimize web architecture before deployment which leads to better adoption, reactive fixes and costly late-stage rework.

B. DL-BASED APPROACHES

1) Paper 1 (DL): Automated Performance Testing Based on Active Deep Learning [17]

This study introduces ACTA (Active learning for Test Automation), which is a framework for automated performance testing of web applications. It leverages techniques such as deep learning, active learning, and conditional generative adversarial network (CGANs). Unlike traditional approaches that require exhaustive test generation or rely on white box access to system internals, the framework operates in black box settings and autonomously generates test cases that reveal the performance bottlenecks, especially in scenarios where there is limited test budgets and large input spaces. The core idea of ACTA is a CGAN-based model, trained to generate positive tests. The logic of this approach is as follows: the generator produces candidate test inputs, while the discriminator determines whether each test is real or synthetic and aligned with the specified conditions. In order to improve the efficiency of ACTA, uncertainty sampling is selected to choose only t high discriminator uncertainty test for execution on the system under test (SUT). The results of the execution are fed back into the CGAN, leading to an improved ability to generate effective, targeted performance tests. Figure 2 below provides the high-level architecture of ACTA, showcasing its relative loop between test generation, discriminatory evaluation, selective execution, And model refinement. The evaluation of the framework was conducted

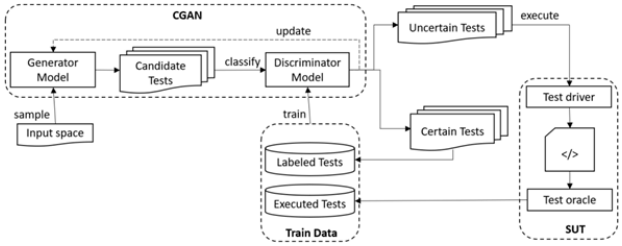

FIGURE 2. Overview of the ACTA Framework [17]

on RUBis, benchmark ecommerce web application. ACTA was compared against 3 baselines: random testing, PerfXRL (reinforcement learning), and DN (discriminative deep learning). The results of these comparisons demonstrated that ACTA is significantly more successful and efficient at identifying bottleneck inducing test cases while also requiring less executed tests. The table below summarizes the number of positive test cases that revealed performance bottlenecks generated by each method across increasing test suite sizes.

TABLE 2. Test-suite sizes generated by each framework

Suite Size	ACTA (active)	ACTA (passive)	DN	PerfXRL	Random
5,000	4,000	4,200	4,300	2,000	1,000
10,000	7,800	7,900	8,100	3,900	2,000
20,000	14,700	14,900	15,300	7,500	3,800
30,000	21,300	21,700	22,400	10,800	5,900
40,000	28,600	29,100	29,800	14,500	7,500

The results of ACTA outperform traditional random and RL-based methods in both efficiency and effectiveness. This showcases the strengths of deep learning powered, active test generation for black box performance in complex web systems. Furthermore, ACTA can be incrementally updated in DevOps environments to adapt to system changes, which makes it more practical and scalable tool for continuous performance assurance.

2) Paper 2 (DL): DeepPerform: An Efficient Approach for Performance Testing of Resource-Constrained Neural Networks [19]

This paper proposes a novel performance testing framework tailored specifically for adaptive deep neural networks (AdNNs) – called DeepPerform. The framework is deployed on resource constrained environments like mobile devices, and unlike traditional DNN testing this approach focuses on

correctness, explicitly input dependent performance bottlenecks (IDPBs), where certain inputs could trigger significant competition overhead and energy consumption without affecting output accuracy. The core idea of this system is to autonomously generate performance test samples that maximize AdNN's competitional complexity, measured in FLOPs. DeepPerform make this happen by training a generative adversarial network (GAN) in order to produce small perturbation on seed inputs for the purpose of increasing resource usage without alternating the semantic meaning. These test cases are then optimized using a multi object loss function, combining adversarial indistinguishability, redundancy amplification, and perturbation boundedness. The generator architecture employs an encoder-decoder with ResBlocks, while the discriminator validates sample realism. The framework is both hardware-agnostic and highly scalable, requiring only forward passes at test time. Figure 3 below introduces the GAN-based architecture used in DeepPerform, which actively learns to generate test cases that maximize activation of neural blocks. In order to evaluate DeepPerform, the others utilized 5 ADNNs trained on CIFAR-10, CIFAR-100, and SVHN datasets. They compared these trained models with the state-of-the-art ILFO method and measured performance degradation using FLOPs, latency, and energy. The results of this was that DeepPerform consistently outperformed ILFO, Both in test generation efficiency and severity of induced degeneration. Table 3 summarizes how DeepPerform achieved higher competition load across all models compared to baseline methods.

TABLE 3. Comparison of FLOPs metrics for baseline vs. DeepPerform-generated cases

Model	Baseline (Mean)	DeepPerform (Mean)	Baseline (Max)	DeepPerform (Max)
SN_C10	6.43	31.14	18.43	62.77
BD_C10	48.44	38.39	162.58	188.60
RN_C100	133.67	181.57	498.29	498.99
DS_C100	116.19	157.66	287.98	552.00
DS_SVHN	115.99	228.32	498.29	498.29

In addition, the study also demonstrated that DeepPerform's Generated input causes up to 552% increase in FLOPs, while maintaining lower generation overhead (6-10 ms per sample). Moreover, DeepPerorm's robustness was showcased across different hardware platforms and AdNN configurations and prove that it's capable of block-level behavioral coverage superior to prior tools . DeepPerform Has proven to be a major step forward in automated deep learning-based performance testing, particularly when put in scenarios in which correctness is not an issue, but computational spikes can cause mission critical failure (e.g drones, mobile devices). Its design allows for scalable and intelligent discovery of worst-case inputs, which can inform both design time optimization and runtime mitigations.

C. RL-BASED APPROACHES

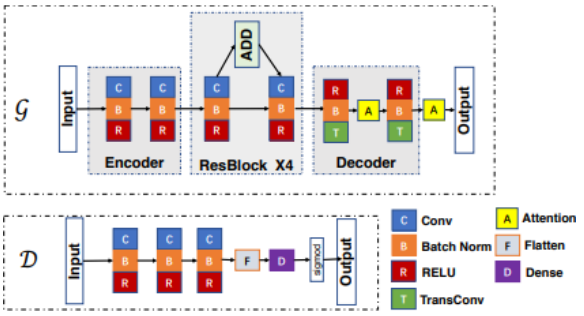

FIGURE 3. Relative performance impact of DeepPerform-generated test cases on AdNN models: percentage increase in FLOPs, latency, and energy consumption compared to the ILFO baseline.

1) Paper 1 (RL): Performance Testing Using a Smart Reinforcement Learning-Driven Test Agent (RELOAD)[3]

This study (2021), proposed framework called RELOAD, a model-free reinforcement learning-driven load testing agent. This model was designed for the purpose of autonomously generating effective workloads for web application performance testing. Unlike the traditional approaches that rely on system models, source code analysis, or static workload generation, RELOAD learns the optimal workload generation policies dynamically through interaction with the system under test (SUT). Its functions entirely under black box conditions, which makes it highly practical for modern software environments where internal artifacts may be unavailable. The RELOAD agent utilizes Q-learning, which is a classical model-free RL algorithm, for the purpose of optimizing the configuration of concurrent transaction loads based on observed system states defined by response time and error rate. At each learning step, the agent manipulates the number of virtual users assigned to the specific web transactions and receives a reward signal based on the progress towards performing test objectives. Through the iterative transaction and policy iteration process, RELOAD learns how to generate finely tuned workloads that trigger performance thresholds violations with minimal test cost. Figure 3 describes RELOAD's high level architecture, showing its integration with a load test actuator (Apache JMeter) and it's a internal learning and decision making loop based on state action pairs. The evaluation of this model was conducted on a real-world ecommerce web application deployed using Woocommerce. A comparison between RELOAD and standard baseline and random testing was performed across several learning configurations including variations of -greedy exploration and a DQN-based variant.

Moreover, RELOAD demonstrated better transfer learning capabilities, by effectively utilizing learned policies to maintain efficiency across new test objectives without the need for retraining. This adaptive capability makes it specifically suitable for continuous performance regression testing in DevOps environments. Sensitivity analysis further confirmed that proper tuning of learning hyperparameters (e.g., learning

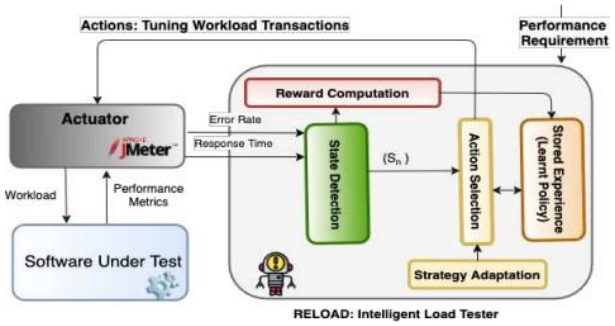

FIGURE 4. Reload testing agent [3]

TABLE 4. Test cost savings under different configurations

Configuration	Vs. Standard Baseline (%)	Vs. Random Testing (%)
$\epsilon = 0.2$	30%	17%
$\epsilon = 0.5$	30%	17%
Decaying ϵ	34%	20%
DQN Setup	34%	20%

rate and discount factor) significantly impacts convergence speed and overall test cost savings. Through the integration of RL techniques into performance testing, RELOAD has proven to be a powerful paradigm for autonomous workload generation that allows for minimal human intervention, system knowledge requirements, and operational costs. This paves the way for scalable, and intelligent testing of complex web applications.

2) Paper 2 (RL): Automatic Web Testing Using Curiosity-Driven Reinforcement Learning (WebExplor)[2]

This paper presents WebExplor, which is an end-to-end automated web testing framework. The way this model works is by leveraging the curiosity-driven reinforcement learning to intelligently explore complex web applications. This method addresses the major limitations of manual, random, and model based testing approaches. WebExplor is capable of autonomously generating high quality action sequences to trigger deep bugs and uncovering hidden states within dynamic, highly interactive web applications. At the core of the WebExplor, the model utilizes Markov Decision Process (MDP). States are abstracted from the HTML structure of web pages, while actions represent operable DOM elements such as buttons, forms, or navigation links. In order to incentivize diverse exploaration, WebExplor adopt a curiosity-driven reward function, which encourages the agent to prioritize transitions that lead to novel states. In addition, this framework builds an incremental Deterministic Finite Automaton (DFA) during the exploration, which offers high level guidance for revisiting promising under-explored paths. WebExplor's architecture consists of three main components: preprocessing for state abstraction, curiosity-based policy learning, and DFA-guided exploration for enhanced efficiency. This is shown in Figure 4 below: This introduced

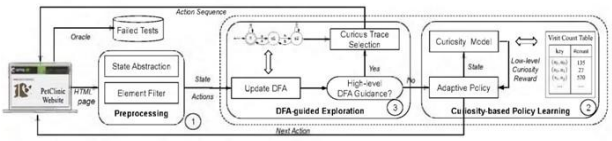

FIGURE 5. WebExplor Workflow [2]

framework was actually evaluated across three different ways:

1) Six open-source web projects using different frontend frameworks (Vue.js, React, AngularJS)
2) A commercial SaaS web platform
3) Fifty top-ranked real-world web applications (per Alexa rankings)

This comprehensive experiment has shown that WebExplor is capable of outperforming the state-of-the-art model-based (DIG, SuWEB) and model-free (Crawljax, Random) testing techniques, in both branch code coverage and failure detection. The number of unique failures detected, and the summary of the average branch coverage are presented in the table below for both baselines across the six research benchmark applications and WebExplor.

TABLE 5. Branch coverage for WebExplor vs. best baselines

Subject	WebExplor Coverage (%)	Best Baseline Coverage (%)
Dimeshift	51.0	40.1 (SuBWEB)
Pagekit	36.0	31.7 (SuBWEB)
Splittypie	43.4	46.9 (DIG)
Phoenix	81.7	65.1 (SuBWEB)
Retroboard	61.4	71.9 (SuBWEB)
PetClinic	85.0	83.0 (WebExplor)

In addition to the high coverage, WebExplor discovered 3466 unique failures across real world applications, many of which involved critical server-side and client-side errors (e.g., JavaScript exceptions, server 500 errors, security exposures). Furthermore, in the commercial SaaS case study, WebExplor uncovered 12 previously unknown failures, all confirmed and fixed by the developers. By integrating curiosity-driven exploration with DFA-based high-level guidance, WebExplor sets a new benchmark for intelligent, scalable, black-box web testing, demonstrating that RL techniques can effectively overcome the exploration challenges posed by complex modern web systems.

IV. COMPARATIVE ANALYSIS

Machine learning, Deep learning, and Reinforcement learning, each of these approaches offer distinctive advantages and trade-offs when it comes to web application performance testing. Machine learning techniques, such as support vector

TABLE 6. Comparison of ML, DL, and RL–Driven Web Testing Approaches

Aspect	Machine Learning (ML)	Deep Learning (DL)	Reinforcement Learning (RL)
Main Contribution	Predicts web app performance based on static attributes	Generates adversarial/performance-degrading inputs	Dynamically explores and generates test cases based on interaction
Data Requirement	Structured metrics and technical features	Large, labeled datasets or simulations	Direct real-time interaction with the web app (black-box)
Strengths	Fast training; simple, interpretable models	Captures complex, non-linear patterns	Autonomous and adaptive; no system model needed
Limitations	Poor handling of dynamic interactions	High compute and data demands	Slower learning curve; sensitive to reward design
Testing Phase Focus	Early-stage performance prediction and design guidance	Stress testing and bottleneck discovery at runtime	Real-time exploration; uncovering hidden failures
Best Use Cases	Predicting performance before deployment	Testing robustness under dynamic conditions	Intelligent exploration of complex interactive systems

machines and random forests, are quite effective for early-stage performance prediction. This is done by leveraging metrics like page load times, usability score, and technical optimization parameters, ML are quite capable of predicting potential performance issues during the design phase rapidly. The key strength of these techniques is in the simplicity, speed of training, and interpretability, making them suitable for proactive performance assurance. However, they are often faced with limitations in handling the dynamic and stateful nature of real-world web applications, specifically when it comes to complex user interactions influencing performance outcomes. Deep Learning methods surpassed the capabilities of classical ML, this is done by modeling nonlinear and high dimensional relationships within data. Frameworks such as ACTA and DeepPerform showcased that DL can autonomously generate inputs to expose the worst-case scenarios without the need for prior system models. DL-based methods excel in specific conditions such as stress testing dynamic systems where potential variations in input can lead to significant performance degradation. Nevertheless, these applications come with challenges as well which include high competitional cost, data dependency, and longer model training time. Lastly, reinforcement learning, which is presented by frameworks such as RELOAD and WebExplor, offers a completely different approach by interacting and exploring the web application under test. RL agent learns the optimal way and strategy to test through a phase of trial and error, adapting their exploration policies based on observed system responses such as response time and errors. This intelligent dynamic exploration allows RL systems to uncover deep hidden failures that the other static models or scripted tests may miss. The main limitation that RL approaches face includes lower learning curve, hyperparameter sensitivity, and the need for careful designed reward function to guide effective exploration. Overall, the selection choice between ML,DL, and RL approaches should be based on objectives and context. It's clear that ML is best suited for early prediction and design time optimization, while DL is a powerful tool for running stress testing and bottleneck detection, while RL excels in autonomous, black blocks exploration of complex and dynamic web application where system internals are unknown.

V. RESEARCH GAPS AND CHALLENGES

The applications of machine learning, deep learning, and reinforcement learning techniques has managed to advance web application performance testing significantly. However, several critical challenges and research gaps yet remain. First, data availability and quality are considered a major limitation. Obtaining large, diverse, and representative datasets in real world development environments hinders the model training and generalization. Second, that generalization across architectures is limited. When applying the models to different systems, they often struggle due to the fact that they're trained on specific stacks such as microservices, and monoliths. This reduces their usability and scalability. Third, interoperability remains a problem, especially when dealing with DL and RL approaches. These approaches often act as a "Black Boxes" and obscure the reasoning behind detected anomalies. Moreover, resource constraints include the high computational cost of DL models as well as RL agent interaction overhead. These problems restrict the adoption of AI based methods in resource limited settings. In this specific case of RL models, reword design continues to be a challenge as poorly constructed rewards can misguide exploration and reduce testing efficiency. Addressing these gaps by developing lightweight, interpretable, and generalizable AI-driven frameworks presents a promising direction for future research.

VI. CONCLUSION AND FUTURE WORK

This review paper analyzed AI driven techniques that use machine learning, deep learning, and reinforcement learning in web application performance testing. Some approaches, such as classical supervised models allowed for early-stage performance prediction based on structured metrics. While DL methods leveraged neural networks as a powerful tool for stress testing dynamic systems and uncovering perfor-

mance bottlenecks without explicit system models. While on the other hand, RL strategies such as curiosity-driven and model-free agent enhanced autonomous exploration and intelligent workload generation under black box settings. The comparative analysis done in this study highlights that each AI paradigm has distinct advantages depending on what stage of testing and the system complexity. It was observed that ML is best suited for design time performance estimation, while DL excelled in runtime stress testing, and RL was the most effective in dynamic, real time system exploration. However, facing challenges such as data availability, model generalization, interpretability, resource demands continue to limit the broader adoption of AI driven performance testing frameworks. Future research should concentrate on implementing lightweight, generalizable, and explainable AI-based performance testing methods. The focus should be placed on hybrid approaches that combine the strengths of the three different techniques as well as the creation of public benchmarks to allow for standardized evaluation. In addition, integrating continual learning and transfer learning techniques into AI-driven performance testing could potentially enhance adaptability fast evolving web application environments.

VII. REFERENCE

[1] H. Wei, J. S. Rodriguez, and O. N.-T. Garcia, "Deployment Management and Topology Discovery of Microservice Applications in the Multicloud Environment," J Grid Comput, vol. 19, no. 1, p. 1, Mar. 2021, doi: 10.1007/s10723-021-09539-1.

[2] Y. Zheng et al., "Automatic web testing us- ing curiosity-driven reinforcement learning," in Proceed- ings – International Conference on Software Engineering, IEEE Computer Society, May 2021, pp. 423–435. doi: 10.1109/ICSE43902.2021.00048.

[3] M. H. Moghadam et al., "Performance Testing Us- ing a Smart Reinforcement Learning-Driven Test Agent," in 2021 IEEE Congress on Evolutionary Computation, CEC 2021 – Proceedings, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 2385–2394. doi: 10.1109/CEC45853.2021.9504763.

[4] J. Huggins and S. Contributors, "Selenium-web browser automation," http://www.seleniumhq.org.

[5] M. Biagiola, A. Stocco, A. Mesbah, F. Ricca, and P. Tonella, "Web test dependency detection," in Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, New York, NY, USA: ACM, Aug. 2019, pp. 154–164. doi: 10.1145/3338906.3338948.

[6] A. Stocco, R. Yandrapally, and A. Mesbah, "Visual web test repair," in Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, New York, NY, USA: ACM, Oct. 2018, pp. 503–514. doi: 10.1145/3236024.3236063.

[7] S. Thummalapenta et al., "Efficient and change- resilient test automation: An industrial case study," in 2013 35th International Conference on Software Engi- neering (ICSE), IEEE, May 2013, pp. 1002–1011. doi: 10.1109/ICSE.2013.6606650.

[8] Google, "Monkey," https://developer.android.com.

[9] "Crawljax," https://github.com/crawljax/crawljax.

[10] M. Biagiola, A. Stocco, F. Ricca, and P. Tonella, "Diversity-based web test generation," in Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, New York, NY, USA: ACM, Aug. 2019, pp. 142–153. doi: 10.1145/3338906.3338970.

[11] A. Mesbah, A. van Deursen, and S. Lenselink, "Crawling Ajax-Based Web Applications through Dynamic Analysis of User Interface State Changes," ACM Transactions on the Web, vol. 6, no. 1, pp. 1–30, Mar. 2012, doi: 10.1145/2109205.2109208.

[12] S. Athaiya and R. Komondoor, "Testing and analysis of web applications using page models," in Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis, New York, NY, USA: ACM, Jul. 2017, pp. 181–191. doi: 10.1145/3092703.3092734.

[13] Y. Zheng, Z. Wang, X. Fan, X. Chen, and Z. Yang, "Localizing multiple software faults based on evolution algorithm," Journal of Systems and Software, vol. 139, pp. 107–123, May 2018, doi: 10.1016/j.jss.2018.02.001.

[14] "ISO/IEC 25010, System and software quality models, ISO 25000 Series," https://iso25000.com/index.php/en/iso25000- standards/iso-25010.

[15] Z. M. Jiang and A. E. Hassan, "A Survey on Load Testing of Large-Scale Software Systems," IEEE Transactions on Software Engineering, vol. 41, no. 11, pp. 1091–1118, Nov. 2015, doi: 10.1109/TSE.2015.2445340.

[16] N. Kaushik, H. Kumar, and V. Raj, "Micro Fron- tend Based Performance Improvement and Prediction for Microservices Using Machine Learning," J Grid Comput, vol. 22, no. 2, Jun. 2024, doi: 10.1007/s10723-024-09760-8.

[17] A. Sedaghatbaf, M. H. Moghadam, and M. Saadatmand, "Automated Performance Testing Based on Active Deep Learning," in Proceedings – 2021 IEEE/ACM International Conference on Automation of Software Test, AST 2021, Institute of Electrical and Electronics Engineers Inc., May 2021, pp. 11–19. doi: 10.1109/AST52587.2021.00010.

[18] L. Kumar, S. Tummalapalli, S. C. Rathi, L. B. Murthy, Krishna, and S. Misra, "Machine learning with word embedding for detecting web-services anti-patterns," J Comput Lang, vol. 75, Jun. 2023, doi: 10.1016/j.cola.2023.101207.

[19] S. Chen, M. Haque, C. Liu, and W. Yang, "DeepPerform: An Efficient Approach for Performance Testing of Resource-Constrained Neural Networks," in ACM International Conference Proceeding Series, Association for Computing Machinery, Sep. 2022. doi: 10.1145/3551349.3561158.

[20] M. Ghattas, A. M. Mora, and S. Odeh, "A Novel Approach for Evaluating Web Page Performance Based on Machine Learning Algorithms and Optimization Algorithms," AI (Switzerland), vol. 6, no. 2, Feb. 2025, doi: 10.3390/ai6020019