GUI Testing using Reinforcement Learning

Sahithi Muppiri   
*Computer Science MS  
Purdue University Fort Wayne*  
Indiana, USA  
mupps02@pfw.edu

Rakshya Aryal   
*Computer Science MS  
Purdue University Fort Wayne*Indiana, USA  
aryar01@pfw.edu

Manpreet Singh   
*Computer Science MS  
Purdue University Fort Wayne*  
Indiana, USA  
singm06@pfw.edu

*Abstract*— *To make sure that the user interface works according to the intended function and to reduce the hustle of manual testing, a deep reinforcement learning approach is introduced in this paper. It is assumed that this approach will be a better way to test the graphical user interface and to ensure that the test suites adapt to any changes in the system without the need to manually feed data. Software testing aims to speed up testing operations while also ensuring that the discovered software bugs can be replicated. The paper discusses a solution to automate the user events such as determining what the user clicks next based on the Q-Learning algorithm to GUI testing, which discusses how the user interacts with the system. Using deep reinforcement learning, a new technique will be provided to automate the test cases. The test cases will also be developed and run manually as well as automatically and will compare the results with the deep reinforcement approach. In addition to that, all the system states are discovered, and possible testing sequences are also determined by the deep reinforcement learning model. The automated or manual testing agent begins by traversing the tested environment to determine the most effective paths to maximize the coverage while finding bugs. For example, if testers could concentrate more on the functionality testing to increase the complete quality of a software and ensure to have maximum code coverage which would be the focus in the paper. The algorithm over the course learns to explore more efficiently as the random algorithm and more efficiently than an untrained user using this new way. We will be testing the developed model on a few scenarios and evaluate the performance among different autonomous testing procedures*

Keywords—reinforcement learning, GUI testing, automated testing, manual testing, user interface, bugs

# Introduction

Software testing is about checking whether a software product matches the expected requirements. In addition to that, it also ensures that software products are error free. In other words, the main use of software testing is to find the missing tests, identify errors within a source code, gaps or missing functionalities with respect to actual requirements. So, it is critical to test software during the development process. It also helps to assure its quality. It is usually done using two different approaches i.e., manual, and automated. Regarding the types, based on what feature of software is to check as well as how it is done, there are mainly two types of software testing viz. Functional testing, and non-functional testing. Functional testing is about testing the functionality of the software under test. Smoke testing, integration testing, regression testing, unit testing, localization testing are some examples of functional testing. There are some other aspects that are important for the performance of the software like performance of the application, time, accuracy, durability and so on. They fall under non-functional testing. Both types of testing are equally important and play a crucial role in developing a good, and reliable software product.

As discussed above, the software testing is an important phase in the software development process since it ensures the quality and performance of software applications. Software testing not only ensures a reliable product but also helps management to get the idea of the work that has been completed. It has been a long practice in the market that human testers conduct manual tests, by going through each stage of a test case and making sure that it performs as intended. For this, the testers click links and input values to the input fields, press buttons and then get the output. They verify that the inputs are working as per customer's requirement and the software is giving the correct output. They then report the bugs and any issues found to the development team so they can fix them. It is the simplest form of testing, very easy to implement and very close to real user feedback. But at the same time, it may not detect all defects as the testers only test what has been listed or asked, takes a very long time, might include human error and is mostly not reusable. However, with automated testing, the processes and verification are scripted using test scripts. These test scripts test the software and validate the program automatically. Even though manual testing has been the predominant method of testing for many years, automated testing has recently gained popularity. Manually doing testing with complicated GUIs and regular releases is tedious, time-consuming, and resource-draining. Most automated testing approaches now in use rely on script execution for testing. These scripts can be performed several times, making the testing process faster and easier. Random testing, in which the testing tool creates unsystematic, random actions based on data obtained from the software under test(SUT), is used in some other automation attempts. Manual testing, on the other hand, is adaptable because it is done by testers, whereas automated tests must be rebuilt whenever the software is changed. This encourages the development of new adaptive and automated testing methods.

In this paper, a new machine learning algorithm has demonstrated the ability to learn from examples and generalise to previously unknown data will be discussed. This study proposes a fresh approach to GUI testing automation that uses deep reinforcement learning[1],[2]. Many of the most effective machine learning techniques are supervised or unsupervised.

Reinforcement Learning (RL) is a machine learning method for determining the best control strategy for agents interacting with an unknown environment. RL is a technique for figuring out how to get to a given desired state to maximise a numerical reward signal, or how to map situations to actions based on interactions with the environment. In RL, agents attempt to analyse an action and make the next decision based on observations of the environment's response. Our method teaches the machine how to interact with the GUI, evaluate which GUI state to investigate first, and take steps to transition from one state to the next. As a result, by exploring as many unexplored environment states as possible, we may increase coverage and detect any potential problems or crashes. Because they may learn directly by interacting with software, reinforcement learning approaches are suitable for software testing. There are so many other benefits of reinforcement learning like it can correct the errors that occur during the training process. That means the model gets familiar with the way the programmer writes the code and the possible errors that can occur with that particular way of coding. So, once the error is fixed by the model, there will be very less chance for the same error to occur again. In addition to that, this technique can be used to solve the problems that cannot be solved by conventional techniques, and it is preferred for long term results. In other words, it is more or less similar to teaching the human being and it can learn from its own experience even if there is no training dataset. But it is not the perfect way and there are some negative sides as well. Some examples of such cons are that it may not be very useful and efficient for simple problems, it needs a lot of data to train it properly and so on. Keeping both advantages and disadvantages in mind, this is a great technique, and its pros weigh a lot more than its cons. It also has a lot of applications in the current software development and testing market.

Diagram

Description automatically generated

Figure 1. Q learning algorithm

In this paper, the popular Q-Learning algorithm is used to automatically parameterize all the web elements  for backend and frontend through a reinforcement learning technique that memorizes an action-value function and then follows a present strategy. One of the most important features of QLearning is that it can compare expected rewards without the need for a prior knowledge of the environment.

This algorithm uses the Bellman’s equation in order to record the to get the value of a state which is equal to the maximum of different rewards that you can get from the state by performing any n actions, R (s, a) and the discounted value of a new state where you will land upon by taking that particular action ‘a’. [13]

Bellman Equation -> **V(s) = max(R (s, a) + γV(s’) )**

The results from the Q learning algorithm are fed into the QTable which is a general lookup table that calculates the maximum expected rewards from the algorithm for each action that it takes at each state.

Table

Description automatically generated

Figure 2. A sample Q Table

The exploration and problem detection in this paper are focused on web-based user interface. Web based include both the Frontend and the Backend parts of web applications. The user interface and logic for handling user activities such as button clicks and interactions with the backend are found in the frontend. The frontend receives the data from the backend, which it stores in the database.

The remainder of this paper is laid out as follows. Next section will talk about the manual testing approach applied on one webpage for the comparison. It talks about how it was done and what result was obtained while performing the test manually. Section II gives a quick review of autonomous testing procedures used at several domains. The scripts were written using the selenium framework and tested on the login page of the university student portal. That shows how the test scripts can be automated and also helps to compare this approach with manual testing as well as the new reinforcement learning approach. The testing domains usually include the web-based user interface testing. In section III it explains what are the problems that will arise when we automate exploratory software testing with deep reinforcement learning. This begs the question: can deep reinforcement learning be used to replace the requirement for manual testing in user interface testing?

# related work

At the initial stages of using test cases to test the graphical user interface. User actions need to be recorded, which is why manual testing played an important role until recently. The abundant time, cost and human resource required is quite complex. The necessity to automate the testing of the user interface to help build on the quality of the product became the most prioritised requirement. Monkey testing framework is one such automated testing method to test the GUIs, it involves randomly clicking or interacting with GUI elements and observing how the tested application behaves[6], for example, to check if the UI has crashed. Monkey framework also comes with some useful features which includes checking if a guided action sequence is needed based on the current GUI view. This is the case with login dialogs since they require specific usernames and passwords to be bypassed [2].

Next comes the autonomous testing which includes random testing, one of the testing techniques. Tools like monkey runner, dynodroid are used to test android applications. This is a command line tool which can run on any emulator device or mobile [7]. It uses a script or randomly streams several user events like click, touch etc., and monkey runner is used to stress-test the applications in a random but repetitive fashion. This is one of the effective approaches as during its execution when it catches any errors or bugs, it terminates and reports the error.

Several possible testing approaches have been implemented in the past to improve the performance the testing approaches.

## Model Based Testing

This is a form of a testing approach that forms the basis of mental development of the subsequent development of the models by a learner. These models often interact with each other and produces a dynamic phenomenon. There are four model-based learning algorithms which uses a range of uses if predictive models. [14]

1. Analytical gradient computation
2. Sampling-based planning
3. Model-based data generation
4. Value-equivalence prediction

Where the above methods discuss several methods that could be used to incorporate reinforcement learning methods.

In the simple model-based testing approach without the reinforcement learning method, the test cases are generated from the given application based on the functionality and application flow. This particular testing approach help us to expediate the testing and will help the developers and also the testers to create a better software.

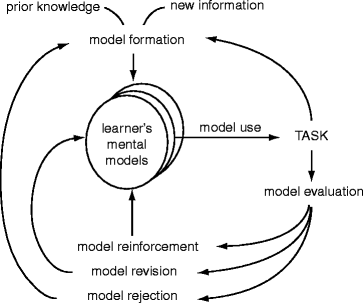


Figure 3. Model-based testing flowchart

## Record and replay

This is one of the codeless automation testing techniques. One way this could be very helpful in GUI testing is that the tester doesn’t need any programming language to run the tests.

These tools are initially developed to fit the organizations with a best test automation strategy. Companies that practise the agile methods cannot use this kind of testing approach as a very small change in the workflow of the website requires the tester to record the content of the webpage again.

These might not be very feasible for the large organizations as there will be continuous change in the website due to the changing requirements of user based on the current trends. Also, more complex coding methods would be required as this approach is a straightforward one. But in advantage is that many record and replay tools can help to cover both mobile and also web user interface to complete test coverage when the platform is not changed.

## Workflow Guided Exploration (WGE)

The other autonomous testing technique is the use of Workflow Guided Exploration (WGE). This is a testing technique procedure which targets using the actual GUI execution on the SUT to help make the agent learn quickly. The WGE approach consists of two modules: replay module and capture module. The Replay module, as by its name helps to reproduce the events. However, the capture module monitors user interaction with the GUI. Which raised to another feature, record and replay testing tool. Since many frontend applications require app-specific programming or recording actions [5], SWTBot, a record and replay tool, is introduced. This uses replay technique to improve the accuracy in discovering SUT bugs. Record and replay tools may not be totally effective since the recording can be useful only if there is no change in the application, which cannot be possible as every app requires updating. Applying any changes to the UI, we won't be able to use the recorded actions and would need to re-record, which is not efficient.

*C. Q-Learning*

The Q-learning approach to automate the UI testing. This method uses the reinforcement learning technique to automate the testing process. This addresses the pitfalls in the previously mentioned methods for GUI testing by enabling high code coverage results unlike random testing. This paper focuses on the Q-Learning approach to improve the testing accuracy.

Regarding the literature, paper [11] talks about the application of deep reinforcement learning for augmented automated game testing. Gaming industry is one of the rapidly growing areas of software development and it has limitless possibilities of improvement and testing to make a competitive commercial gaming software’s. Just like their features, they demand extensive testing during the development. Manual testing might not be very useful in these sorts of applications and also the automated testing might not be efficient as there will be thousands of possible cases that might need to be tested. In addition to that, the testers might need extensive gaming knowledge to produce the relevant test data. So, this paper introduces a self-learning mechanism to the game testing environment. It claims that, with their deep learning reinforcement learning, their framework is capable of exploring and/or exploiting the game mechanics based on a user-defined, reinforcing reward signal. The result of this is increase in the test coverage and it also helped figure out the bugs.

The article [12] explains the major things to consider while implementing ML/AI in test automation. It says that automation does the job but using artificial intelligence along with machine learning for testing take the overall practice to the next level. Overall, the machine learning in software testing makes the test creation process faster and easier, makes the test analysis simpler and also reduces the test maintenance.

# problem

Although we have several automated testing tools discussed over the section II. As discussed above, each of the testing method has its own advantages over the others as well as they have drawbacks too. Concentrating on either of the drawback a new testing method is born after it. Manual testing procedure requires a lot of human effort and time. Which is why automated testing coming into picture. However, using automated tools like monkey framework has certain limitations, in this tool, it is very difficult to find the unexpected issues found during the testing. Testers can’t exactly define the accurate test cases. Addressing the drawbacks, WGE testing techniques are introduced. However, this method can’t be available for larger demonstrations. Record and replay tool which partially addresses the solution, yet when there is a change in the application, we need to re-record all the events which is time consuming and requires a lot of human effort.

In addition to that, due to today's technological advancement, the application and the platform for the software have been increasing a lot and the need for software testing is growing rapidly. For example, the same application should be able to work on smartphones, PCs with various operation systems as well as IOT environments. This leads to the wide variety of testing on various platforms and numerous integrations. Since smart solutions are used everywhere, there are a lot of use cases in many domains like gaming, automotive, industries, robotics, healthcare, education and many other areas and the application is growing like never before. This increase in application domain requires vast knowledge of that particular field. Also, that is causing the generation of huge amounts of data. The more data we create and the more applications we explore it brings more challenges for software developers. Also, a lot of knowledge is required to test successfully, and the learning never ends. But if a self-learning environment can be developed that can test maximum possible test cases itself, the overall testing process will be simpler, easier, less time consuming and cost effective.

Deep Q-learning approach resolves the issues in either of the abovementioned testing procedures, yet in this case a discrete environment is required to find the best action from a given current state. Although this method provides good code coverage and find bugs and/or crashes in the software. But what happens when the users restrict over certain actions? To evaluate if the proposed solution checks the logical errors. To determine the time taken to run the test suite.

# Solutions

A GUI is a graphical user interface that shows objects that communicate information and reflect user actions. When the user interacts with these objects, they change in colour, size, and visibility. However, GUI testing can be quite challenging. To think what the user will do, to verify if the user can understand what is given. In simple terms to think from the user perspective is quite a task.

We divided the problem statement with the various approaches that could be used to measure the performance of the test from each of the method. These approaches are:

* Manual Testing
* Automation Testing
* Q Learning Algorithm

To compare the effectiveness of each of the approach, a simple shopping cart website is chosen for GUI testing. There are several websites that we found online, which could be used for testing the automation processes. One such website is called as the sauce demo which is a sample user interface that lets you to login and purchase products that are available online. It lets you to add the products to the cart and remove them from the cart. User can enter his home address and contact information and then it will navigate to a page regarding the product confirmation.

Graphical user interface, application

Description automatically generated

Figure 4. Graphical User Interface for testing

There are four possible web pages that can be used by the user within the given website. One is the login page, which can be used by any person to login from the possible combination of username and a single password that is given at the bottom of the login button.

Once the user enters the combination of the data to login, next web page is displayed within the same page, as a single page application, the page then displays a list of few products in a grid which consists of products image with a button in the end of each image which lets the user to add the product that they want to purchase. He is given an option to select as many products as he can and each time, he selects an option an increment of one is added to the cart symbol at the top right corner of the same page.

After choosing the products the user can select the cart icon which navigates to another page. At this page he can view the products that he has chosen and then remove the item that he no longer needs.

Later when he decides the products, he can select the checkout option and then enter the details of where the items need to be shipped to. Since this is a sample web page it doesn’t ask you any personal details and no information that you provide is saved in the database. After entering the details, there is a button in the bottom that allows you to book the order for the products that you choose. Then a conformation message is displayed on the screen after successful completion of the steps that the website is designed for. A full advantage of this website is used for the project

Few of the possible GUI actions that our testing approaches test are:

* Login and password, text fields are tested if they are writable.
* When the used enters an invalid credentials, there is a pop-up menu that shows an error message, that the details are wrong and asks to re-enter.
* When a valid credentials are used, then he is allowed to enter the product information page, where he can choose the products.
* The dd button is validated if products are added to cart and then an increment on the cart is shown simultaneously.
* When user selects cart icon on the top right, a new page should be opened.
* If user stays more than few minutes on the same page, then the user is automatically navigated to the logout page.

The following actions are tested using each of the approach mentioned above. There are other actions like the text fields if they are placed on the page with a proper alignment for this when the website is opened using any developer tools provided by browsers for instance, Google Chrome we can inspect all the elements of the website for example text fields, buttons, labels etc., Inference can be taken from this approach.

Graphical user interface

Description automatically generated

Figure 5: Inspecting the Graphical User Interface

But there is a limit to this method since all the web pages can’t be used to cross check with this approach. Since the website can be accessed between multiple platforms like Mozilla Firefox, Google Chrome, Microsoft edge in different devices like tablet, mobiles, desktops etc., with different resolutions and different screen sizes. This drawback can be addressed in the future by using any effective artificial intelligence approach.

We discussed how the test can be conducted on each of the testing approaches.

I.I Manual Approach

The very first testing we considered is Manual Testing. This is a very basic test which is very simple and straight forward. There is no learning curve in this kind of approach. A basic template is designed for this purpose where the user can assign a testcase ID for each of the test that he conducts on the user interface. Initially the user has to study the complete user requirements and enact himself as a user and find possible drawbacks from the website designed.

To implement this type of testing, a software tester must understand the functioning of the website too. He needs to verify that the user doesn’t spend more than few minutes to think what he needs to do in a website.

A template is a simple document that can be compatible in any system. Our template includes the product title or the page that we are currently testing. The name can be in a simple, understandable three-word text at most, the version of the web page that we are currently testing is also given, a short description of the web page, a possible or predicted release date along with seven columns of test cases are written.

Graphical user interface, application, table, Excel

Description automatically generated

Figure 6. Manual Test Cases

Tester writes all the possible that can be conducted on the page. He validates then and records the output for each and sends for a review.

I.II Automation Tools Approach

Another approach is to automate instead of human spending so many hours to test a website. To do this, a tool is chosen which helps in automating the test cases and is used to test the graphical user interface.

a. Selenium Tool

1. Automated testing tool that is free and open source.
2. Supports a variety of operating systems and web browsers
3. It's well-known for web application testing.
4. It provides regression testing, recording and playback options.

One of the best advantages of this tool is that it allows every programming language to work using the benefits of this environment. Selenium suite contains various tools such as IDE, Web Driver, Client API, Remote Control and Grid.

b. IDE

Visual Studio Code is used in used to integrate the selenium framework and automate the testing process. This beautiful IDE is developed by the Microsoft corporation. It supports several languages like C++, Ruby, HTML, CSS, bootstrap, Fortran, JavaScript and many more. This tool can be used in any operating system like the Windows, Apple mac, Linux and ubuntu environments. The reason for its popularity is that the several functionalities that this tool offers. Few of them are code refactoring, debugging the code, allowing to use the version control system, GIT, syntax highlighting, customizable theme or the interface, ability to install several packages and extensions.

C. Packages

1. Chrome Driver

WebDriver is a free, open-source tool for testing web applications across a variety of browsers. It can navigate to web pages, accept user input, execute JavaScript, and so on. Chromed river is a stand-alone server that adheres to the W3C WebDriver specification.

2. Pandas

This is a Python module for data science, data analysis, and machine learning. It's based on NumPy, a multi-dimensional array support package. Pandas, as one of the most widely used data manipulation packages, integrates effectively with a variety of other Python data science modules.

Automation testing was implemented on the same website that was used for manual testing using selenium in VS code software. Different items were tested like navigation(links), error messages, required input fields, and the functionality of GUI elements were tested and verified that all those elements work as intended and were able to do the complete testing within 8 secs. The following figure shows the flow of the test performed:

Diagram

Description automatically generated

Figure 7: Flow of automation script

III. QLearning Approach

Q-learning is one of the reinforcements learning algorithm which trains itself from the possible combination of input. This is one of the effective approaches to determine the outcome of the next state based on the learnings. It trains itself and evaluates the decision accordingly.

a.      IDE

Visual studio Code is used in q learning approach. This is chosen over the other IDE’s due to the simplicity of the interface and the compatibility of several packages that we would use over the project.

b.      Framework

Selenium testing framework is used to further extend the autonomous testing process. This tool is used in the automation testing as well. The testing method seems to be effective as it uses several

c. Packages

We would require several packages in order to automate the GUI.

|  |  |
| --- | --- |
| **Package/Libraries used** | **Description** |
| NuGet Package | This is a global package folder which can install any downloaded packages. Each of the package if expanded into subfolders which match the package identifier and a number |
| Xunit | This is a free source unt testing framework that is focused on unit testing the tool. |
| WebDriver | This web driver is used to create a browser-based regression automation test suites and tests. This is used to enable the introspection and control of the user agents. |
| netstandard2.1 | This is a formal specialization of .NET API which are available on multiple implementations. This is used to make sure that the code can be portable in any environment possible. |

Table 1. Packages, libraries used

1. Xunit.

This is an open-source testing tool, which is used for unit testing. This is used in the .NET Framework. This is a tool that allows you to custom develop new attributes which is used to control your tests. It ensures that the functionality with the possibility of extending the class’s like DoesNotContain, Contains, InRange & NotlnRange etc.,

# Results

The main purpose of testing using different approaches was to compare and evaluate the drawbacks of one testing method and try to resolve them in the other approach. We found that the manual testing can be quite time consuming, and the tester cannot always cover all the testcases or can found all the errors from a website.

In order to save the huge time complexities, automation testing approach is used which can be used to feed the data and helps us to find out all the possible missing testcases that could be covered. Then comes the artificial intelligence approach, Q-Learning. The major goal here is to show how the model could be used to find bugs and crashes while also enhancing testing coverage.

|  |  |
| --- | --- |
| **Method** | **Time (in seconds)** |
| Manual Testing | Can range based upon the platform that we test and human effort. |
| Automation testing | 8 seconds to run 10 tests. |
| Qlearning | 9 seconds to run 15 tests by training itself. |

Table 2. Time complexity based on different approaches used.

The results of the tests are measured and expressed using a variety of metrics. This approach uses an effective system where the path that has not been covered by the system can be covered. Another factor is the average unique states per action, which is calculated by averaging all distinct states for each action.

# Conclusion

Traditional approach of manual testing is quite time consuming, and it might vary according to the human requirements to design a website. If the website os pretty simple and is used for a very small applications without any change in the software, then we can choose the manual testing approach.

The next approach that we choose is by using the automation tool to test the selected user interface as in manual testing. So here we observed that the tool when the script is correctly written for the software without any changes in the user interface, then the tool runs in less than few minutes.

Using Q-learning, this work proposes a reinforcement learning technique to automate GUI testing. The suggested technique is based on the exploration of software by a Q-learning agent to boost testing coverage and detect bugs and/or crashes. The tool's evaluation revealed that it achieved good testing coverage, presenting an innovative technique in the field of autonomous GUI testing. The parts that were not covered by the tools were proved as defects or unwanted code area that needs to be evaluated.

# Challenges

Comparing and observing the several testing approaches, though certain process can be effective and yield best solutions while testing yet one solution cannot be always effective in the testing process.

There are certain challenges that we found over the course of the project discussion; few such are:

1. Error handling can be quite complex when the project is so large
2. When the User interface keeps on changing then the automated tests has to be fed with the environment and when reinforcement approach is used, then the tool needs to learn the entire process again.
3. The pixel size in the image keeps on changing over the generations. Handling of the UI can be crucial with varying size, color, shape and maintaining scripts can be quite challenging.
4. Increasing test complexity in the modern applications have complex features embed into them like the frames, maps, flowcharts etc.,

The above-mentioned challenges can be addressed in the future using the artificial intelligence with better high performance computing machines. This is required as the tool needs to be learned again and again to understand the environment. Several cycles of test are executed for the tool to understand the environment. If the website is added with the new changes too, then the tool can detect the changes and train itself.

# future work

Deep Q learning approach can be used with the Q learning algorithm. The huge difference among the wither of them comes while handling the rewards in the Q table. Deep Q learning algorithm can replace the traditional Q learning table with the neural networks. Instead of mapping the state-actions pair to a q-value, with the neural network we can maps the input states to actions and a Q-value pair.

To increase testing coverage levels, more complex tools that enable a wide test coverage can be used. Currently, there are no constraints to the training process, and it is performed on the entire program.

Also, the test cannot be generic. That is if a test has been conducted on a machine and when few states are already tested, when the test is triggered again, it doesn’t need to test all the states again. Instead, it can be used to test the uncovered paths and test the ones that doesn’t provide a good result while testing.

In addition, more crash and/or problem states for the system to detect would be good.

# References

[1] Saber, S., Elbadry, F., Negm, H., El-Ershad, R. A., Magdy, O., Bahnassawi, M., ... & Bayoumi, A. (2021, December). Autonomous GUI Testing using Deep Reinforcement Learning. In *2021 17th International Computer Engineering Conference (ICENCO)* (pp. 94-100). IEEE.

[2] Juha Eskonen, Julen Kahles and Joel Reijonen, “Automating GUI testing with image-based deep reinforcement learning”, August 2020. In 2020 IEEE International Conference on Autonomic Computing and SelfOrganizing Systems (ACSOS) (pp. 160-167). IEEE.

[3] J. Kim, M. Kwon and S. Yoo, "Generating Test Input with Deep Reinforcement Learning," 2018 IEEE/ACM 11th International Workshop on Search-Based Software Testing (SBST), 2018, pp. 51-58.

[4] Eskonen, J. (2019). Deep Reinforcement Learning in Automated User Interface Testing.

[5] T. Wetzlmaier, R. Ramler, and W. Putschogl, “A Framework for Monkey ¨GUI Testing,” in 2016 IEEE International Conference on Software

Testing, Verification and Validation (ICST), April 2016, pp. 416–423.

[6]“MonkeyRunner” https://developer.android.com/studio/test/monkeyrunner

[7] Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi and Percy Liang, “Reinforcement Learning on Web Interfaces Using WorkflowGuided Exploration”. Department of Computer Science, Department of Statistics Stanford University, Stanford, 24 Feb 2018.

[8] “SWTBot” http://wiki.eclipse.org/SWTBot/UsersGuide

[9] T. Vuong and S. Takada, “Semantic analysis for deep q-network in android gui testing,” 07 2019, pp. 123–128.

[10] Joakim and S. Takada, “Semantic analysis for deep q-network in android gui testing,” 07 2019, pp. 123–128.

[11]Joakim Bergdahl, Camilo Gordillo, Konrad Tollmar, Linus Gisslen´ SEED - Electronic Arts (EA), Stockholm, Sweden - Augmenting Automated Game Testing with Deep Reinforcement Learning

[12] Prathutha Lyengar, headspin-  Machine Learning in Test Automation:6 Things to Be Considered.

[13] <https://blog.floydhub.com/an-introduction-to-q-learning-reinforcement-learning/>

[14] https://link.springer.com/referenceworkentry/10.1007/978-1-4419-1428-6\_589