A Case Study of Testing an Image Recognition Application

Chuanqi Tao①②, Dongyu Cao①, Hongjing Guo①, Jerry Gao③

① College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China

②Ministry Key Laboratory for Safety-Critical Software Development and Verification, Nanjing University of Aeronautics and Astronautics, Nanjing, China

③Department of Computer Engineering, San Jose State University, USA

Correspondence to: taochuanqi@nuaa.edu.cn

***Abstract*—High-quality Artificial intelligence (AI) software in different domains, like image recognition, has been widely emerged in our lives. They are built on machine learning models to implement intelligent features. However, the current research on image recognition software rarely discusses test questions, clear quality requirements, and verification methods. This paper presents a case study of a realistic image recognition application called Calorie Mama using manual and automation testing with a 3D decision table. The study results indicate the proposed method is feasible and effective in quality evaluation.**

Keywords-image recognition; testing AI software; AI software quality validation

I. INTRODUCTION

With the rapid development of big data analysis and artificial intelligence technology, AI software and applications have been widely accepted in our daily life. At present, AI software and applications are based on the most advanced machine learning models, and various artificial intelligence features are realized through large-scale data training.

The most important implementation of Artificial Intelligence is the imitation of human interactions—vision. Nowadays, there is an abundance of digital images captured by high-quality equipment. Most images are captured with phones. Artificial Intelligence is often used to process these images to extract knowledge, categorization, and labeling along with other advantages. Typical applications of image recognition include object recognition, face recognition, text parsing.

Detecting bugs and errors in software can be very costly. Sometimes bugs can be even deadly if it is a real-time application of software, such as some software that is used to help with surgeries in the hospital. Therefore, testing the software is very important to verify that the product meets requirements and specifications. Software testing ensures the correctness, integrity, and high quality of the software by checking errors or bugs and fixing them in the initial design.

This paper focused on testing an image recognition application called Calorie Mama utilizing both manual testing and automation testing. Calorie Mama is a smartphone app that runs on Android and IOS devices. It uses deep learning to recognize food from food images and track nutrition based on the food in the image. It calculates the calorie based on that. We evaluated the performance, correctness, and quality of the app using both manual testing and automation testing.

This paper is written to provide our perspective views on image recognition software testing and quality evaluation. The paper is organized as follows. Section 2 discusses the review of AI software testing and image recognition. The third part shows a case study of testing Calorie Mama APP using manual testing and automation testing. Section 5 gives the conclusion finally.

II. RELATED WORK

Traditional software is implemented by developers with carefully designed specifications and programming logic. It is tested with test cases that are designed based on specific coverage criteria. However, the current practice of testing AI applications lags far behind the maturity of testing traditional software applications [1].

More and more work focused on testing AI-based software. Gao et al. [2] explained the various testing methods of AI software testing Various functional and non-functional quality parameters such as correctness, reliability, and scalability are discussed to better understand the concepts. Besides, the authors discussed the issues and challenges of AI testing. The different models of the AI system were discussed in [3]. The authors discussed building testable AI systems, limiting the AI system to propositional logic, and intervening variables in reducing testing. King et al. [4] discussed issues and challenges in software testing. They thought non-determinism is a huge issue. The same input to the system can produce different outputs. Testing has fuzzy oracles that determining the correctness can be a challenging task. Ramanathan et al. [5] used symbolic decision procedures coupled with statistical hypothesis testing to validate machine learning algorithms for studying the correctness of intelligent systems. They also used algorithms to analyze the robustness of a human detection algorithm built using the OpenCV open-source computer vision library.

In the field of image recognition, most of the researchers focus on recognition algorithms. Girshick et al. [6] proposed the R-CNN algorithm, which added selective search operations to the CNN network to identify candidate regions. He K et al. [7] proposed the SPP-Net algorithm, which reduced the process of image normalization and solved the problem of image information loss and storage. Girshick [8] proposed the Fast R-CNN algorithm, which refers to the Region of Interest and the multi-task loss function method, and replaces SVM classification and linear regression with Softmax and SmoothLoss to realize the unification of classification and regression and reduce the disk space.

However, the evaluation of the image recognition system is relatively less but important. In [9], the implementation of Yolo-v2 image recognition and other test benches for a deep learning accelerator was presented. Tao et al. [10] performed a case study

DOI reference number: 10.18293/SEKE2021-194

on a realistic facial age recognition provided by Alibaba Company using metamorphic testing.

III. A CASE STUDY

A. *Test Experiment*

This paper took the test Calorie Mama APP as an example, using manual testing and automated testing respectively. The test data is a mix of various sources: images from Google, images clicked in real-time using a smartphone camera. The experiments were performed with a high-resolution and high-quality camera.

(1) Manual Test

In this approach of manual testing, we selected conventional decision tables to test. A decision table is a table with various conditions and their corresponding actions. It is divided into four parts, condition stub, action stub, condition entry, and action entry.

1)Detection of non-food items: To test Calorie Mama, different non-food items are input into the application. The pictures of the non-food items were analyzed by the application and the results were shown on the user interface. A summary of the detection of the non-food items can be seen in the following decision table. The condition stub is designed as two conditions, including the state of the Internet and access to the Camera, which is essential for the image recognition software.

As we can see, the application detected artificial pumpkin and artificial cake as food items. In contrast, it could not correctly identify the butter block. As a result, it failed in some of the cases. Besides, when not turning on WIFI or Cellular, and not allowing access to the Camera, image recognition will not work.

TABLE I. DECISION TABLE OF THE NON-FOOD ITEMS

Test Conditions	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**
Turn on WIFI or Cellular	T	T	T	T	T	T	T	T	F
Allow access to Camera	T	T	T	T	T	T	T	T	F
Food item	**Pen**	**Apple**	**Artificial Pumpkin**	**Butter Block**	**Banan a**	**Chicken Wings**	**Clarified Butter**	**Artificial Cake**	**Glass of Water**
Detected as food	F	T	T	F	T	T	T	T	-
Not detected as food	T	F	F	T	F	F	F	F	-

2)Detection of food items: We divided the generic term of food items into four categories which are Indian cuisine, raw fruits and vegetables, variety of apples and eggs, and food items in different backgrounds. Take food items in different backgrounds as an example, the background of food is a very important aspect and we decided to test the application with images of food items with different backgrounds.

As seen in table 2, the Calorie Mama application was able to correctly recognize the food items when given inputs with red, blue, and wooden backgrounds. However, the application detected wrong when the egg is in a tray.

TABLE II. DECISION TABLE OF FOOD ITEMS IN DIFFERENT BACKGROUNDS

Test Conditions	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**
Turn on WIFI or Cellular	T	T	T	T	T	T	T	T	T
Allow access to Camera	T	T	T	T	T	T	T	T	T
Food item(Egg)	**Blue Back-ground**	**Red Back-ground**	**Wooden Background**	**Egg in a bowl**	**Egg on a plate**	**Egg on a pan**	**Egg in the glass**	**Egg in a jar**	**Eggs in a tray**
correct choices	T	T	T	T	T	T	T	T	F
wrong choices	F	F	F	F	F	F	F	F	T

After conducting the manual testing, we experienced its various drawbacks, and it is time-consuming. Also, load testing and performance testing are not possible under manual testing. Besides, regression test cases are very costly. Due to these drawbacks, we decided to shift to automation testing.

(2) Data Modeling

The three-dimensional (3D) classification decision table is influenced by the concept of conventional decision tables to conduct classification-based test requirement analysis and modeling for any given mobile apps powered with AI functions using a 3D tabular view. The major testing focus for a 3D classification table is the mappings among classified disjoint context conditions, classified input selections, and classified AI function outputs. These mappings are known as image recognition function classification rules. Each of them represents the conjunction among three different views. Test case design and generation based on a 3D classification decision table must cover these image recognition classification rules. Adequate image recognition function testing coverage could be assessed. Next, we introduce the construction of each one-dimensional model in the 3D decision table.

1) Input Modeling

The input classification refers to the parameters and their values that represent the different test case scenarios. Each parameter has multiple possible values which when combined with context values gives us the final set of test cases. The following figure shows Calorie Mama's input classification tree, which contains information about the type of food being clicked, such as what the food is, and the physical appearance of the food, such as quality, size, shape, consistency, etc.

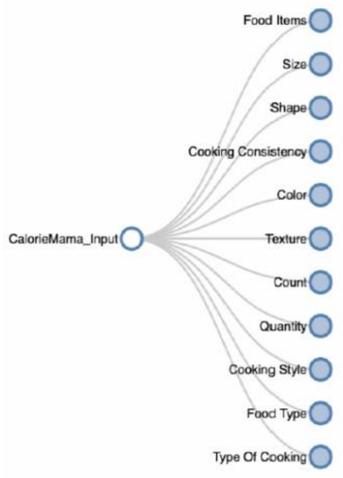

Figure 1. Input Classification Tree

2) Context Modeling

The context classification tree contains information about the image context. It is basic information about the image itself and not specifically about the item in the image. For example, the context classification tree contains information like if the image is blurry or not well illuminated, what is the angle of the camera while clicking the image, if the image is rotated or so, etc. The following figure shows Calorie Mama's context classification tree.

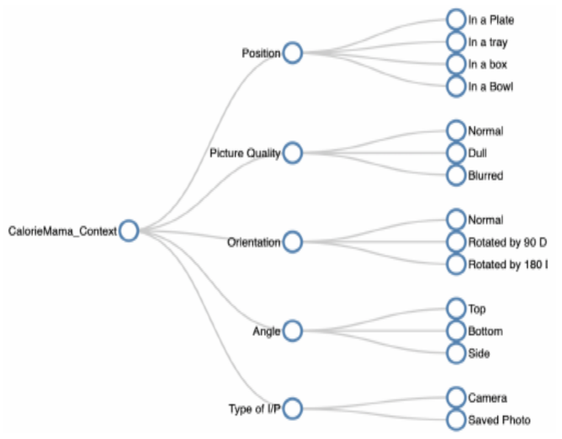

Figure 2. Context Classification Tree

3) Output Classifications

The output classification tree contains information about the output. Various parameters regarding the output obtained from the application will be considered. This can be modified based on the requirements and results expected from the application. The following figure shows Calorie Mama's output classification tree.

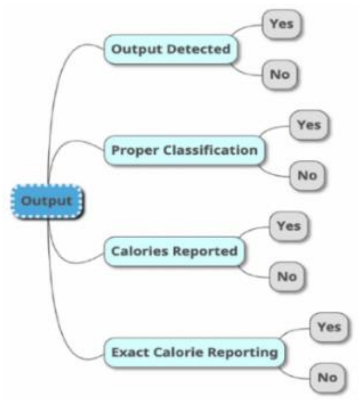

Figure 3. Output Classification Tree

(3) Automation Test

After data modeling, we performed automation testing with minimal human assistance on top of the model. Automation testing can increase coverage for test data and come up with more concluding test results for the selected mobile app. We used Appium as an automation tool to perform automation on the mobile app. Appium acts as a server that launches the app into the simulator or a real device and can access the elements for processing the actions triggered by the automation script which we wrote in Java. Steps to perform the automation were:

1. Install Appium server.
2. Create the automation environment for Android.
3. Create the automation environment for iOS.
4. Launch simulator/ Connect a real device.
5. Install Eclipse.
6. Create a maven project in Eclipse to write and run the automation script.

We provide the dependencies of Appium, Selenium, TestNG in the Project Object Model and then start writing the scripts. We use TestNG to run our automation tests. Soon after the execution of tests, test results are visible in the Eclipse console.

For the algorithm of the app automation, one image which is selected from the gallery of the phone is fed as an input into the target app, and the result of the execution is compared with the expected output. If the output from the target app is as expected, then the test case is displayed as passed or else failed. Also, when the app produces the output, more options, as provided by the app are taken into account. While showing the output to the user, there is an option to see more options from the suggestions coming from the app. The algorithm considers all those options as the output from the app and then decides if the test case is passed or failed.

B. Test Result

After applying manual testing and automation testing, we compare the coverages for both manual and automation tests. In manual testing, the coverage of the test case was limited due to timing. It was difficult to cover a larger set of data without the use of tools or scripts. On the other hand, automation testing has higher coverage because the tools and script helped us to cover more test cases. Figure 4 below shows that in automation testing we were able to cover more test sets of data than the manual testing over the same time. Approximately, in the automation testing, we were able to cover twice of what we covered in the manual testing.

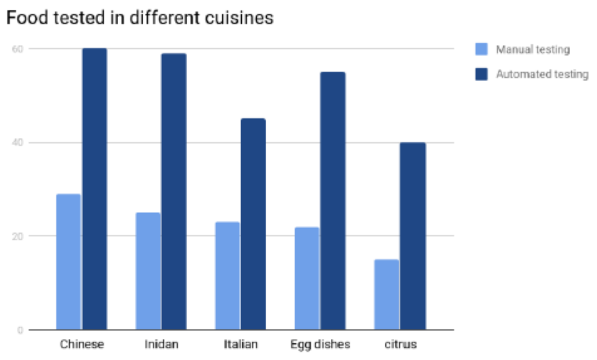

Figure 4. Test Coverage for Manual and Automation Test

The app was able to detect objects, recognize them, and classify them with its name. However, it does not tell the count or sub-classification of the food item. Moreover, testing Calorie Mama App, required a lot of time to do both manual testing and automation testing. Manual testing needs to take more time to generate all decision tables, analyze different test causes and test manually. On the other hand, in automation testing, we spend days to get the script working correctly and program it to do the testing automatically.

The following figure shows the results of the manual testing and automation testing of the Calorie Mama APP. In manual testing, the total test food item across different cuisines was 400 items and each cuisine has 80 food items. The 132 of them were wrongly detected they were bugs in the app. This gives us a 33 failed percentage and the passing percentage is 67. The diagram below shows the failing and passing results.

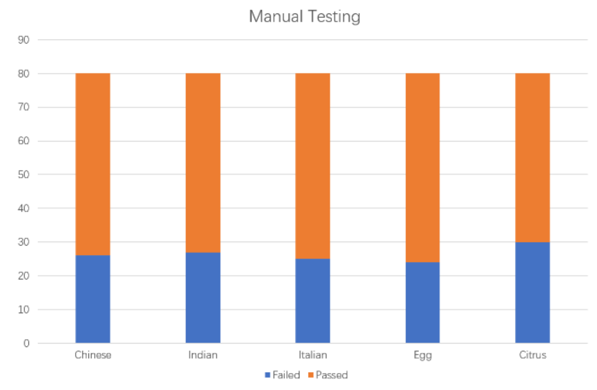

Figure 5. Manual Testing

In Automation testing, we tested 400 different images in different cuisines similarly. We found out that out of the 400 images, 175 failed and 225 passed. This gives us a failure percentage of 43.75 and a passing percentage of 56.25 as shown in figure 6.

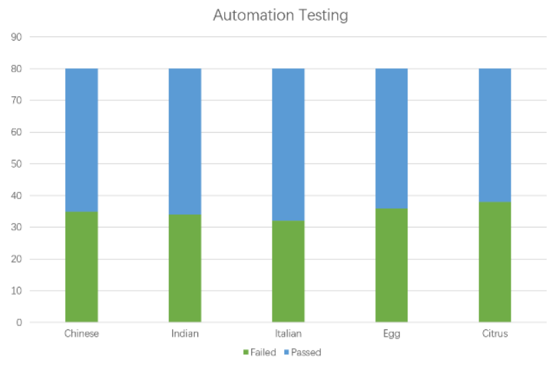

Figure 6. Automation Testing

Comparing the manual testing with automation testing, we can see that the errors that were found in the automation testing are higher than the errors that were found by the manual testing because the automation test allows us to test different inputs in a short time. Also, in manual testing, it is more likely to make human mistakes because doing repeated tasks over time generates more errors by humans. However, doing a repeated test using automation by writing a script and let the machine discover the error is more efficient. Therefore, automation testing discovers more errors than manual testing.

IV. Conclusion

To sum up, we mainly leverage two methods to test the image recognition system, namely manual testing, and automation testing. In manual testing, the test is conducted by human testers inputting the use cases one by one, and observing the results. Manual testing can be expensive and time-consuming. Moreover, it is subject to human error; therefore, it is not one hundred percent accurate. On the other hand, in automation testing, the testers use tools and scripts to help them conduct the test among the image recognition software, which can save labor and time cost, thus improving testing efficiency. It helps them find errors without the need of performing redundant tasks. However, it needs talented and experienced people to do that, which is expensive. Besides, it is difficult to automate all kinds of testing where not everything can be redundant and reusable.

Acknowledgment

This work is supported by the Foundation of Graduate Innovation Center in Nanjing University of Aeronautics and Astronautics (kfjj20201603).

References

[1] H. Zhu, D. Liu, I. Bayley, R. Harrison, and F. Cuzzolin: Datamorphic Testing: A Method for Testing Intelligent Applications. In: IEEE International Conference On Artificial Intelligence Testing (AITest), pp. 149-156. Newark, CA, USA (2019).

[2] Gao, J., Tao, C., Dou, J., Lu, S., 2019, “Invited paper: What is AI software testing? and why,” 13th IEEE International Conference on Service-Oriented System Engineering, SOSE 2019, San Francisco, CA, USA, April 4-9, 2019, pp 27-36.

[3] G. Liu, Q. Liu and W. Zhang, “Model-based testing and validation on artificial intelligence systems,” Second International Multi-Symposiums on Computer and Computational Sciences (IMSCCS 2007), Iowa City, IA, 2007, pp. 445-449.

[4] T. M. King, J. Arbon, D. Santiago, D. Adamo, W. Chin and R. Shanmugam, “AI for Testing Today and Tomorrow: Industry Perspectives,” 2019 IEEE International Conference On Artificial Intelligence Testing (AITest), Newark, CA, USA, 2019, pp. 81-88.

[5] A. Ramanathan, L. L. Pullum, F. Hussain, D. Chakrabarty and S. K. Jha, “Integrating symbolic and statistical methods for testing intelligent systems: Applications to machine learning and computer vision,” 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, 2016, pp. 786-791.

[6] R. Girshick, J. Donahue, T. Darrell and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, 2014, pp. 580-587.

[7] K. He, X. Zhang, S. Ren and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," in IEEE Transactions on Pattern Analysis and Machine Intelligence,2015,37(9):1904-1916.

[8] R. Girshick, "Fast R-CNN," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 2015, pp. 1440-1448.

[9] C. Kim et al., "Implementation of Yolo-v2 Image Recognition and Other Testbenches for a CNN Accelerator," 2019 IEEE 9th International Conference on Consumer Electronics (ICCE-Berlin), Berlin, Germany, 2019, pp. 242-247.

[10] C. Tao, J. Gao and T. Wang: Testing and Quality Validation for AI Software–Perspectives, Issues, and Practices. IEEE Access 7, 120164-120175 (2019).