An Intelligent Mobile Application Testing Experience Report

Chi Thi Phuong Tran¹, Mohana Gudur Valmiki¹, Guoyan Xu² and Jerry Zeyu Gao*¹

¹San Jose State University, USA
²Hohai University, China
*Email: jerry.gao@sjsu.edu

Abstract. Artificial intelligence applications provide tremendous opportunities to improve human life and drive innovation. AI systems/applications which operate in a real-world environment have to encounter an infinite set of feasible scenarios. Conventional testing approach to test the AI application allows only limited testing and does not allow taking the different contexts into consideration and may lead to insufficient validation and characterization. Therefore, to ensure robustness, certainty and reliability of AI applications, the authors applied classification-based AI software testing framework and 3D decision tables to generate test cases. Moreover, the authors compared the quality assurance metrics (accuracy, correctness, reliability and consistency) of AI and non-AI functions in the AI mobile application scenario. Our results indicate and confirm that complete AI function validation is not possible with conventional testing methods, but AI software testing strategy proposed based on classification framework and 3D decision tables has a good effect.

1. Introduction

Recent advancements and development in Artificial Intelligence (AI) and ML (Machine Learning) are greatly influencing our everyday life. Market forecast by Tractica [1] suggest that annual global AI software revenue is forecast to grow from $9.5 billion in 2018 to $118.6 billion by 2025. Implementing the AI and ML methods or technologies in mobile application development is becoming widespread to achieve intelligent functions, such as recommendation functional features, detection and recognition function, Natural Language Processing (NLP), question and answer functions, unman-controlled vehicles. Therefore, it is very important to test them adequately and assure its safety.

AI testing can be categorized into two kinds a) AI-based software testing: refers to the leverage and applications of AI methods and solutions to automatically optimize software testing process in test strategy selection, test generation, test selection and execution, bug detection and analysis, and quality prediction; b) testing AI software: refers to activities related to validating the AI system functions and features that are developed based on machine learning models, establishing AI function quality test requirement, detecting the AI function issues, limitations and quality problems. This paper focuses on the testing of AI software.

Testing of AI mobile applications poses many challenges and conventional methods of testing are not enough to establish quality and assurance of AI applications and functions. The major problems of AI application testing are due to a) Lack of well-defined and experience-approved AI system validation models and methods for AI applications developed based on big data and using machine learning and deep learning techniques; b) Lack of well-defined quality assurance standards and assessment methods; c) Lack of efficient and cost-effective automatic quality validation tools for machine learning based AI systems.

Major challenges include but not limited to a) how to identify and establish quality assurance and
testing coverage criteria requirements for AI functions; b) what are the adequate quality assessment and validation criteria for AI functions; c) how prepare clear and effective problem/bug report and conduct problem/bug analysis for AI functions; d) how can we use systematic methods to generate quality training data sets and test data sets.

For the above-mentioned problems and challenges, the authors proceeded with classification-based AI software testing [2] for AI function inputs, contexts, and conditions to assure the adequate testing coverage. And in this paper, the authors are aiming at errors/bugs discovered using conventional testing methods to implement and analyze by classification-based AI function testing method.

2. Related Work

2.1. AI-Software Testing

The real beginning of AI in the modern world has its foundation in the Turing test introduced by Turing as an “Imitation Game” in 1950 [3, 4], which opened new doors for the AI field [5].

AI-software testing can be defined as testing activities with intent of finding errors in AI-based software using well-defined quality validation models, methods and tools. Its main objective is to validate system functions and features based on AI and ML models. The main obstacles of testing AI applications can be divided into four parts by Li L, et al [5]. a) Detailed description of tasks which can be quantitatively validated; b) How to make sure that the AI application is reacting accordingly to all possible tasks in the scenario. It can be straightforward if few variables exist in the case of simple intelligence but in complex intelligence tests this can be problematic and if there are continuous variables generated [6]; c) How to make the simulation-based test as “real” as possible. Simulation-based tests are an advantage because of higher cost and effort when compared to practical real-time tests but the question is how can we simulate the complex behaviors of certain animals and humans; d) How to establish the appropriate test performance evaluation indices for tasks. Performance indices are difficult to come up with because each human reacts differently, criteria can be complex and moreover, we expect machines to perform better than humans. In testing the intelligent vehicles [5], the authors give the definition and generation of intelligence test tasks for vehicles to combine the benefits of scenario-based testing and functionality-based testing approaches based on a semantic relation diagram definition for driving intelligence [7]. The authors applied the parallel learning method [8-12] to the vehicle intelligent test and proposed a parallel system framework that combined the real-world and simulation-world for the test.

In conclusion, conventional methods, strategies and tools do not accomplish the requirements of AI-software testing.

2.2. Classification Model and Decision Table

Classification-based AI software testing, in which classification models for inputs, contexts, and outputs and events are set up for AI software testing to assure the adequate testing coverage of diverse input data classes, classified contexts and conditions, and corresponding outputs and classes.

To leverage the testing effectively in a short time, the authors applied some basic standard testing methods into some divided testing business domains of the application.

Decision table testing is a kind of testing that is more like a cause-effect testing. This testing will determine what kind of output will be obtained on giving various kinds of input with different kinds of circumstances. Security holes can be detected in this method.

3. AI-Software Testing Modeling Based on Classification and Decision Table

AI technique is applied in many applications that help to reduce human interaction, and then reduce manual working effort. Current testing methods and solutions are not adequate for testing AI software. AI-based software testing cannot be fully effectively executed by existing techniques and tools; therefore, we need to use other quality validation approaches and models that are different from conventional testing techniques and models.
3.1. Mobile Application Scenario
AI software is classified into several systems, such as Expert System, Object Detection & Classification, Business Intelligent System, Behavior Detection & Classification, Identification & Recognition System.

In this paper, the authors selected a testing LookTel Money Reader application that was introduced for enabling people experiencing visual impairments or blindness to quickly and easily identify and count bills [13,14]. It is a kind of Identification & Recognition System. The main features of the application are a) process a video or image and extract the information; b) use patented and proprietary object recognition technology; c) read and identify 21 different currencies; d) voice over support in 17 languages; d) instantly recognizes currency and reads the denomination and displays high contrast large numerals for partial vision loss people.

3.2. AI Function Testing Requirement Analysis

3.2.1. Selected AI Function Features and Requirements. AI function features of checking for capability of recognizing and reading money in conditions are: a) Many equipment types like: many versions of iPhone and iPad; b) One/Many versions of one bill type; c) One/Many banknotes at the same time; d) Noise detection; e) Different lighting conditions; f) Different shooting angles; g) Diverse money conditions (perfect or not perfect currency.); h) Multiple languages/multiple currency types; i) Different distances; j) Moving or Stable state of object; k) Fraud detection; l) Different background where placing bill.; m) Diverse bill color; n) Different bill material.

3.2.2. AI Function Testing Methods with the Necessary Criteria. Without source code provided, the authors had to use Black Box testing with three main techniques: data-driven AI technique, classification and rule-based testing as our testing validation techniques to approach these above features. The authors designed scenarios in which banknotes are selected from some popular/specified currencies in the world to check whether Money Reader can provide exact recognition and read money information in the exact selected language. When testing on AI features, test coverage on all above defined AI functions is done by fulfilling all test cases that will be created in the next section of this paper. After doing some detailed analysis, the authors applied data-driven AI technique, classification testing and rule-based testing to create detailed test cases and perform testing.

3.3. AI Function Testing Modelling
AI testing is executed with some inputs under specific circumstances or context, and will result in some consequences. At the first step of AI testing, the authors will analyze and define the context, which data will be input and finally output of testing.

3.3.1. Context Modeling. The authors divided the context in which the reading of money is executed in some specific conditions of equipment type, equipment position, currency type, state of object (people taking bank note), distance between equipment and bill, and environmental factors like: lighting and background placing bills. In which equipment type contains iPhone, iPad and Mac laptop. Currency type is a group of 4 main language systems: Naga language system, Middle Semit language system, Kanji language system and Latin language system. We defined the distance between the equipment and bills into 3 segments based on our empirical experiment: < 3 cm, > 48 cm, between 3 cm and 48 cm. Figure 1A represents the details of AI context modeling. The notation SELECT-1 means only one option can be selected, SELECT-M allows the combination of multiple conditions. XOR means one of the two options can be selected.

3.3.2. AI Function Input Classifications. We separated the input data into 2 domains: bill state and bill classification as represented in Figure 1B and 1C. Bill state classification describes the various states of the bill used in testing such as, perfect condition, bill with missing parts, shaded bill and finally whether they are in what kind of physical condition such as folded, flat, scratched and wet conditions. Bill
classification describes whether the currency used is bill or coin, fraud bills, and currency made up different material.

3.3.3. **AI Function Output Classifications.** Based on the diverse inputs of currency that are described above and execution in different contexts, the authors classified the output into two major categories that are represented in Figure 1D.

3.4. **AI Function Classification Decision Table**

With all classified spanning trees of AI context, AI inputs and outputs above (3.3.1, 3.3.2, 3.3.3), the authors-built decision tables for each classification that are prerequisites for building test cases.

![Figure 1A. AI context spanning tree.](image1)

![Figure 1B. Bill State spanning tree.](image2)

![Figure 1C. Bill classification spanning tree.](image3)

![Figure 1D. AI output classifications.](image4)
3.4.1. **AI Context Decision Table.** Table 1 shows a random combination of all parameters that are described in Figure 1A, in which “T” means: “True” and “F” means “False”. The total number of rules: \(2^{17}\) (2\(^{\text{number of conditions}}\), in which we excluded 2 conditions of equipment type Dell laptop and Mac laptop due to money reader application cannot be installed on these two machines).

3.4.2. **Bill State Decision Table.** The next decision table represents the various states of the bill, which was developed based on the bill state context classification diagram as shown in Figure 1B. The bill state decision table was a random combination of all options. The authors selected 7 conditions and the total number of rules are \(2^{7} = 128\) rules.

3.4.3. **Bill Classification Decision Table.** The final table represents the different bill classification decision table was developed with bill classification context classification diagram as shown in Figure 1C. The authors selected a random combination of parameters described in Figure 1C, such as authentic, fraud, currency material type of the bill and/or coin to create the decision table. In this decision table the total number of rules are \(2^{12} = 4096\) rules.

| AI Context Decision Table | Rules |
|---------------------------|-------|
| **Equipment**             |       |
| Ipad                      | F     |
| Mini                      | F     |
| Pre                       | F     |
| Air                       | F     |
| Iphone                    | F     |
| S                         | F     |
| Parallel                  | F     |
| Ruptile angle             | F     |
| Acute                     | T     |
| No gap                    | T     |
| **Currencies**            |       |
| Supported Indian rupee    | T     |
| Unsupported Nepal rupee    | T     |
| Supported UAE Dirham      | T     |
| Unsupported Qatar Royal    | T     |
| Supported Yuan            | T     |
| Unsupported GBP            | T     |
| Supported USD             | T     |
| **State**                 |       |
| Stable                    | T     |
| Moving                    | T     |
| **Distance**              |       |
| Too close (<3 cm)         | T     |
| Too far (>48 cm)          | T     |
| **Optimal**               |       |
| **Light Condition**       |       |
| Color light               |       |
| 3-Aiken                   | T     |
| Blue                      | T     |
| Red                       | T     |
| Green                     | T     |
| White light               |       |
| Too dark                   | T     |
| Bright light              | T     |
| Normal light              | T     |
| Low light                 | T     |
| No Noise                  |       |
| **Background**            |       |
| Noisy objects             |       |
| Single object             | T     |
| Multiple                  | T     |
| **Different surfaces**    |       |
| Curved                    | T     |
| Rough                     | T     |
| Plain                     | T     |
| Outputs                   |       |
| Can read the value of bill correctly in selected language & currency type | T |
| Cannot recognize & read the value of bill | T |
| Can read the value of only bill correctly in selected language & currency type, cannot recognize coins | T |

**Table 1.** The AI context decision table.
4. AI-Software Testing Results and Analysis

4.1. Test Results
Figure 2 below represents the total number of AI test cases that the authors implemented with the number of passed/failed test cases and defects found in each business checking. Figure 2B shows the total number of test cases designed using decision table method and scenario test method in each business checking correspondingly. With the definition of 3D decision table, the authors built 21 test cases based on the combination of the first random 21 rules of input and context decision tables.
Figure 2. Illustrate the AI test case results (A: Overall; B: Classification based on test methods).

4.2. Comparison and Analysis

4.2.1. Test Complexity Comparison. To get a general view of testing results, the authors do a comparison between conventional and AI testing in the figures below. Figure 3A consists of a total implemented 55 test cases in which 45 test cases passed and 10 test cases failed with 10 defects found. Meanwhile, 53 test cases passed and 20 test cases failed with 17 defects in AI testing. AI testing techniques work more effectively than Conventional testing techniques in detecting defects. Rate of defects found in AI testing is: 20/73 ~ 27.4%; In Conventional testing, this rate is: 10/55 ~ 19.2%. Besides that, the authors also collect the total number of test cases that were created in each test model using three major test methods and represent them in Figure 3B. In both models, the decision table is the method that is used the most to generate test cases; Scenario testing is used quite similar between AI testing & Conventional.

4.2.2. Bug Comparison. In this experiment, some defects cannot be found in conventional testing but can be uncovered in an AI testing model and vice versa. This result is achieved by using 3D-decision table technique that supports the definition of a wide range of test cases. In Figure 4, we can observe that, in conventional testing we detected 10 defects, in which 3 defects we can uncover in AI testing, but 7 defects cannot be uncovered in AI testing. Meanwhile, 17 defects were found in AI testing, in which 16 defects are not uncovered in conventional testing.

Figure 3. Illustrate the Complexity comparison between conventional and AI ( A: Complexity ; B:Test cases and test methods).
5. Conclusions
In this present work, we focused on the challenges of using conventional software testing methods to test AI applications. The authors tested the currency denomination identifying AI application LookTel money reader using conventional approaches and comparing it with AI function testing. The quality assurance of AI functions is evaluated using metrics such as accuracy, correctness, reliability, and consistency. Our results confirm that indeed conventional testing approaches are not enough to validate the AI functions and using our proposed approach we found 16 defects out of 17 which cannot be found with conventional methods. Finally, we conclude that certainly AI application testing requires a different approach and in future studies should focus on validating the AI application using AI-based approaches so that the entire flow is automated.

Acknowledgement
We would like to sincerely thank Lalitha Rakshitha Raviprasad and Varun Potlacheruvu of software testing group members at SJSU for much work on this paper.

References
[1] Artificial Intelligence Market Forecasts, https://www.tractica.com/research/artificial-intelligence-market-forecasts/
[2] Gao J; Tao C; Jie D; Lu S 2019 Invited Paper: What is AI software testing? and Why. 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE) 2019 27-36
[3] Turning A M 1950 Computing Machinery and Intelligence Mind 59 433-460
[4] Saygin A P, Cicekli I, Akman V 2000 Turing test: 50 years later Minds and Machines 10(4) 463-518
[5] Li L, Lin Y L, Zheng N N, et al. 2018 Artificial intelligence test: a case study of intelligent vehicles. Artificial Intelligence Review 2018(10) 1-25
[6] Lamb E 2016 Maths proof smashes size record: supercomputer produces a 200-terabyte proof—but is it really mathematics? Nature 534(7605) 17–19
[7] Li L, Huang W L, Liu Y, et al 2017 Intelligence Testing for Autonomous Vehicles: A New Approach IEEE Transactions on Intelligent Vehicles 99 1
[8] Zhang N, Wang F Y, Zhu F, et al. 2008 DynaCAS: Computational Experiments and Decision Support for ITS IEEE Intelligent Systems 23(6) 19-23
[9] Wang F Y 2010 Parallel control and management for intelligent transportation systems: concepts, architectures, and applications IEEE Transactions on Intelligent Transportation Systems 11(3) 630-638
[10] Li L, Wen D 2016 Parallel Systems for Traffic Control: A Rethinking IEEE Transactions on Intelligent Transportation Systems 17(4) 1179-1182
[11] Li L I, Lun L Y, Pu C D, et al. 2017 Parallel Learning — A New Framework for Machine Learning Acta Automatica Sinica doi: 10.16383/j.aas.2017.y000001
[12] Li L, Lin Y, Zheng N, et al 2017 Parallel Learning: a Perspective and a Framework *Journal of Automatica Sinica* **4**(3) 389-395

[13] LookTel-Money-Reader Website: http://www.looktel.com/moneyreader

[14] LookTel-Money-Reader Apple store page: https://itunes.apple.com/us/app/looktel-money-reader/id417476558?m