Testing Intelligent Mobile APPs

– An Experience Report and Lessons Learned

**Chi Tran Thi Phuong - chi.t.tran@sjsu.edu**

**Mohana Gudur Valmiki - mohana.gudurvalmiki@sjsu.edu**

**Lalitha Rakshitha Raviprasad - lalitharakshitha.raviprasad@sjsu.edu**

**Varun Potlacheruvu -** [**varun.potlacheruvu@sjsu.edu**](mailto:varun.potlacheruvu@sjsu.edu)

**Jerry Gao –** [**jerry.gao@sjsu.edu**](mailto:jerry.gao@sjsu.edu) **(corresponding contact)**

**San Jose State University**

**Abstract:** Artificial intelligence applications provide tremendous opportunities to improve human life and drive innovation. AI systems/applications which operate on real-world environment have to encounter 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, we applied classification-based AI software testing frame work and 3D decision tables to generate test cases. Moreover, we compared the quality assurance metrics (accuracy, correctness, reliability and consistency) of AI and non-AI functions. Our result show that 16 out of 17 defects found in AI function testing are not exposed in conventional testing, but only 7 out of 10 defects detected in conventional testing cannot be found in AI function testing. Therefore, our results indicate and confirm that complete AI function validation is not possible with conventional testing methods but require different strategies such as classification based, rule-based, model-based, learning-based, AI-based.

**Index Terms** — Artificial intelligence; AI mobile app testing, AI application testing; AI functional testing, AI mobile testing approaches and methods

**1 INTRODUCTION**

Recent advancements and development in the Artificial Intelligence (AI) and ML (Machine Learning) are greatly influencing our everyday life. AI can be defined as, a sub-field of computer science aimed at the development of computers capable of performing intelligent tasks which normally require human intelligence. AI is playing major role in human health, safety, learning and education and increasing the productivity. Moreover, 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 wide spread to achieve intelligent functions such as a) Recommendation functional features in e-commerce and advertising; b) detection and recognition function, for example, human face identification, voice recognition and object detection; c) object identification and classification; d) prediction and business decision-making; d) Natural Language Processing (NLP) capability with language understanding and translation; e) question and answer functions to assist users in messaging, phone calling, search, and smart home appliance control; f) unman-controlled vehicles, robots, and UAVs. Since, AI software or mobile applications are crucial in making important decisions in day to day life, therefore, it is very important to test them adequately and assure its safety. Moreover, testing the AI applications adequately will be helpful in identifying the both desired and undesired effects and necessary corrective actions can be taken quickly.

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.

Testing of AI mobile application poses many challenges and conventional methods of testing are not enough to establish quality and assurance of AI applications and functions. Conventional methods can be employed in system testing evaluating the QoS parameters such as system performance, reliability, scalability, availability, robustness, and security, and etc. Conventional test methods are designed to detect system function errors relating to program logic, business rule and system function based on the limited input data parameters and simple data types. However, most of the AI functions accept various rich media inputs such as text, image, audio and video and generate multiple outcomes. This creates much complexity in test case generation, execution and analysis. Furthermore, conventional testing methods use defined rules to predict the exact output which should fall in certain boundaries. On the other hand, AI functions output might change if the same function is called either in different context or different moments of time as the system will evolve due to learning from the different inputs. Therefore, it is very important to control the input to control the output to an extent.

The main focus of AI application testing are AI functional features to assure their adequate quality in accuracy, consistency, relevancy, timeliness, correctness, and completeness and testing AI software’s quality of system service parameters (system performance, reliability, scalability, availability, robustness, and security) based on well-defined quality standards and assessment criteria.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.

Due to the above-mentioned problems and challenges we proceeded with classification-based AI software testing as suggested previously (2). We used classification-based models for AI function inputs, contexts, and conditions for AI software testing to assure the adequate testing coverage. In this present work we selected LookTel Money Reader (NantMobile Money Reader) application, which provides currency denomination identifier. It works on the principles of Image Recognition and AI. Image Recognition is the ability of software to identify objects, places, people, writing and actions in images. Computers can use machine vision technologies in combination with a camera and artificial intelligence software to achieve image recognition.

In this paper we are aiming at errors/bugs discovered using conventional testing methods compared to classification-based AI function testing of LookTel Money Reader AI functions. We expect that there will be more errors/bugs will be discovered with classification-based AI testing while testing in different contexts

1. **RELATED WORK**

Even though the concept of AI is present in ancient history, the real beginning was started in 1956 and word “Artificial Intelligence” was coined by John McCarthy. The real beginning of AI in modern world has its foundation in the Turing test introduced by Turing as a “Imitation Game” in 1950 [3][4], which opened new doors for the AI field [5].  Turing test takes simple pragmatic approach and showed that computer is indistinguishable from human and machines can think. In brief, the Turing test is a tester would ask the tester freely through some devices (such as a keyboard) in the case where the tester is separated from the tester (one person and one machine). After multiple tests, if more than 30% of the testers are unable to determine whether the tester is a human or a machine, then the machine passes the test and is considered to have human intelligence. Although the Turing test is designed to advance the development of artificial intelligence, it also has several shortcomings such as storage limitations, intelligence boundaries and emotions. The AI research got really picked up after IBM's Deep Blue became the first computer to defeat chess champion Russian grandmaster Garry Kasparov in 1997. Furthermore, IBM’s AI machine Watson won the quiz show “Jeopardy”, a question-answering system against champion Brad Rutter and Ken Jennings in 2011.

Testing AI applications: The main obstacles of testing AI applications can be divided into four parts by Li L, et al [5]. a) Detailed description of task which can be quantitatively validated; b) How to make sure that AI application is reacting accordingly to the all possible task in the scenario. It can be straightforward if few variables exist in the case 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 test 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 react differently criteria can be complex because and moreover, we expect machine to perform better than human. 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].Authors applied the parallel learning method [8, 9, 10, 11, 12] to the vehicle intelligent test and proposed a parallel system framework that combined the real-world and simulation-world for the test.

AI-based testing for AI application: traditional testing strategies and tools do not fulfill the requirements of AI applications. Therefore AI-based testing for AI application is necessary to completely automate the entire AI application testing process. In this kind of approach AI-based testing should automatically provide requirement analysis, test planning, test execution and test analysis report generation of the intended AI application that is being tested. Similar work is reported by Jeremy Straub and Justin Huber [13] using an artificial intelligence test case producer (AITCP) to test artificial intelligence system (AIS). AITCP start from a human-generated test scenario and makes changes to it based upon a modification algorithm such as ant colony optimization and genetic approaches. Comparison of results using manual based and autonomous navigation control system indicated that AITCP can be utilized to effectively test AIS for both surface(two-dimensional) and airborne (three-dimensional) robots.

Detailed description of LookTel mobile app could be found on the website [14] and its download site on the iTunes App store page [15]. According to the vendor the application environment requirement specifications are a) compatible only with iOS 8.0 and higher; b) Required size is 59.2 MB. Moreover, it is also mentioned that the app is compatible with both iPhone and Mac computer (including desktop and laptop).  LookTel application will be greatly helpful in assisting the visually impaired people to recognize the currency denomination and provide voice over support. 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.

Our objective and scope of testing is to test both AI and non-AI features of the application to see whether the application is producing expected output in different scenarios. We will be using only black box testing due to the program is not an open source application. The main categories of AI and non-AI functions that we will be focusing on are a) application setup on various equipment; b) application behavior with currency of different material (plastic, paper, metal etc.,); c) quality of voice over; d) quality of displaying denomination on different types of equipment; e) verify money reader on different currencies that supported and not supported; f) validate the application in various contexts (position, light, distance, angle and environmental factor etc.,); g) validate the application under many conditions of banknote (old, scratched, wet, smudges etc .,).

1. **USING CONVENTIONAL TESTING METHOD**

To leverage the testing effectively in a short time, we apply some basic standard conventional testing methods to this mobile app.

* **Decision Table**

Decision table testing is a kind of testing 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 kind of circumstances. Security holes can be detected in this method. The testing coverage area will be as follow: a) Checking multiple testing condition setup; b) Validate the money reader over some kinds of money material.

* **Scenario testing**

This method is helpful to go through from the beginning state to any states of product without any technical view, only focus on user real experience. In this application, some test cases cannot be created by decision table or equivalence partition because it depends on testing setup and supported features on mobile, not application itself. That’s the reason why we need to use scenario. Test coverage: This method will be used in testing application on different types of supported equipment, with supported features on each, like: voice over setup, voice quality, text displaying, …

* **Equivalence partitioning**

The application involves in detecting the bills from many countries and each country will have different set of bills of different value. This gives us a wide range of input, further which can be partitioned into classes. From each class, a particular kind of input is chosen as given as input. This kind of testing saves the time of the tester and reduces the tester’s workload. Test coverage: This method will help us to cover testing the feature of checking money reader’s detection and voice over capabilities on some range of language system around the world. Base on some defined test methods above, we designed our test cases for each business requirement domain follow each testing technique.

Figure 1 shows total number of executed test cases along with number of pass/failed test cases and number of defects found in each business requirement respectively.

Figure 2 represents total number of testcases designed using decision table method, scenario test method and equivalence partitioning in each business domain correspondingly.

In which, Category 1 = Check installation on supported/un-supported device; Category 2 = Setup the application on iPhone; Category 3 = Setup the application on iPad mini;

Category 4 = Money reader with different types of material; Category 5 = Check quality of voice, text displaying on Category 4 = Money reader with different types of material; Category 5 = Check quality of voice, text displaying on devices’ screen; Category 6 = Check for supported and un-supported currencies.

**Figure 1: Conventional test summary.**

**Figure 2: Summary of conventional test cases and test methods.**

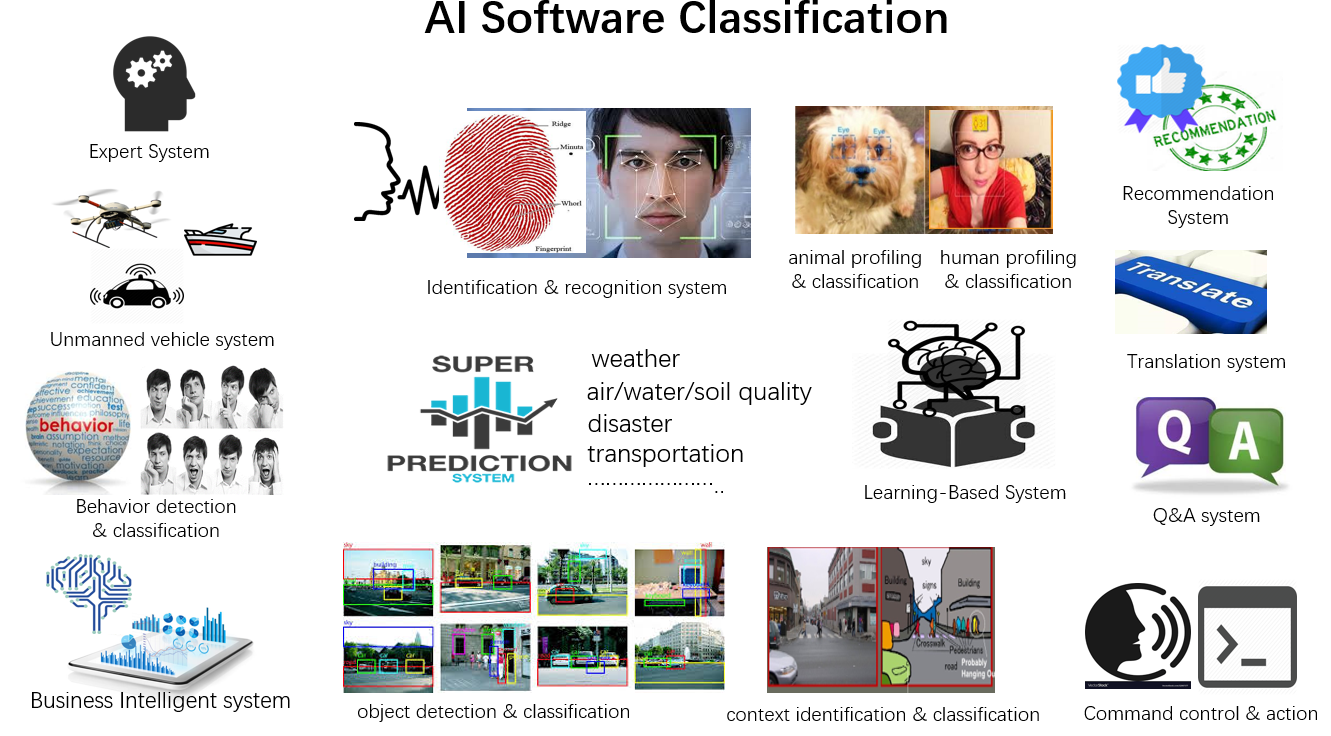
**4 AI** **TESTING, MODELING, and ANALYSIS**

**Why do we need AI testing?**

As we know AI is ubiquitous today. AI technique is applied in many applications that helps to reduce human interaction, then reduce manual working effort. AI is represented for human brain. Current testing methods and solutions are not adequate for testing AI software. AI-based software testing cannot be fully effective executed by existing techniques and tools; therefore, we need to use other quality validation approaches and models that different from conventional testing techniques and models.

AI software is classified into several systems, such as Expert System, Object Detection & Classification, Business Intelligent System, Behavior Detection & Classification, Identification & Recognition System, … conventional testing techniques and models.

AI software is classified into several systems, such as Expert System, Object Detection & Classification, Business Intelligent System, Behavior Detection & Classification, Identification & Recognition System,…The method firstly executes the AI function test modeling from context modeling, test input modeling and test output modeling, and generates a three- dimensional decision table based on the voice input, voice context and the corresponding output and then generate test cases, finally evaluation AI functions quality from three quality parameters of accuracy, correctness and consistency.



**Figure 3: AI Software Classification.**

In this project, we selected testing Money Reader application that introduced for enabling people experiencing visual impairments or blindness to quickly and easily identify and count bills, it is a kind of Identification & Recognition System.

**4.1 AI Function Test Requirement Analysis**

1. **Selected AI Function Features and Requirements**

We will execute AI testing for recognizing and making voice over different types of bills in several languages under

diverse shooting conditions and money backgrounds/conditions.

We will design some scenarios in which banknote are selected from some popular/specified currencies in the world to check whether Money Reader can provide exact recognition and read money information in exact selected language.

AI function features are listed below:

Check for capability of recognizing and reading money in conditions of: 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 materials.



**Figure 4: AI function features.**

1. **AI Function Testing Methods with the necessary criteria**

Without source code provided, we have to use Black Box testing with three main techniques as: data-driven AI technique, classification and rule-based testing as our testing validation techniques to approach these above features.

AI features are the key points of this application; therefore, when we can cover testing on all AI features, we can make sure the application work well. That’s the reason why we will assure to do test coverage on all above defined AI functions by fulfilling all testcases that will be created in the next section of this document.

After doing some detailed analysis, we will apply data-driven AI technique, classification testing and rule-based testing to create detailed testcases and perform testing.

By classifying banknote into several types of currencies with many backgrounds of banknote and putting them under complicated lighting conditions/environment background and can detect whether an AI tool can distinguish all normal and abnormal phenomena on our tested items like human’s eye and brain. In general, we will check money detection base on some inputconditions like: equipment type, equipment position, currency type, state of object (moving or stable), distance from device to bill, environment factor, bill state and bill classification.

**4.2 AI Test Modeling**

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

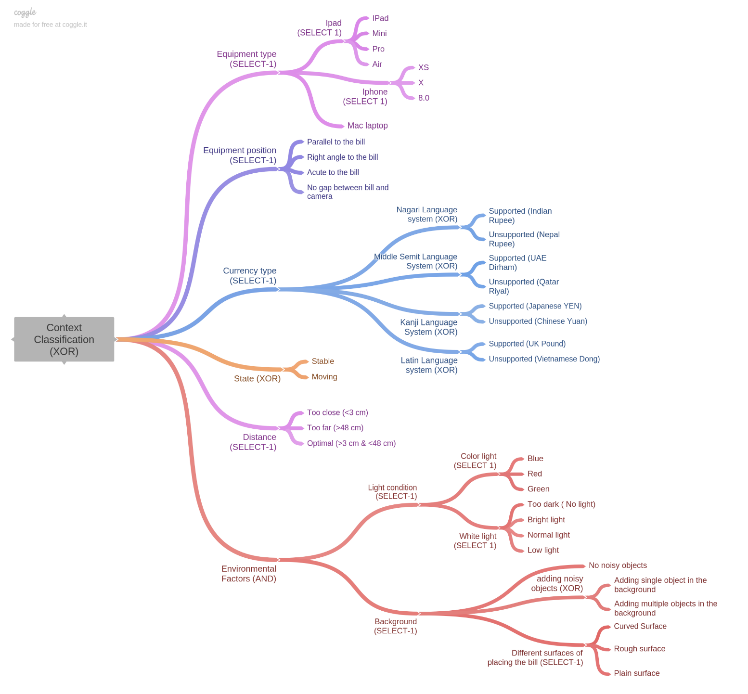
1. **Context Modeling**

We 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 base on our empirical experiment: < 3 cm, > 48 cm, between 3 cm and 48 cm.

Figure 3 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.

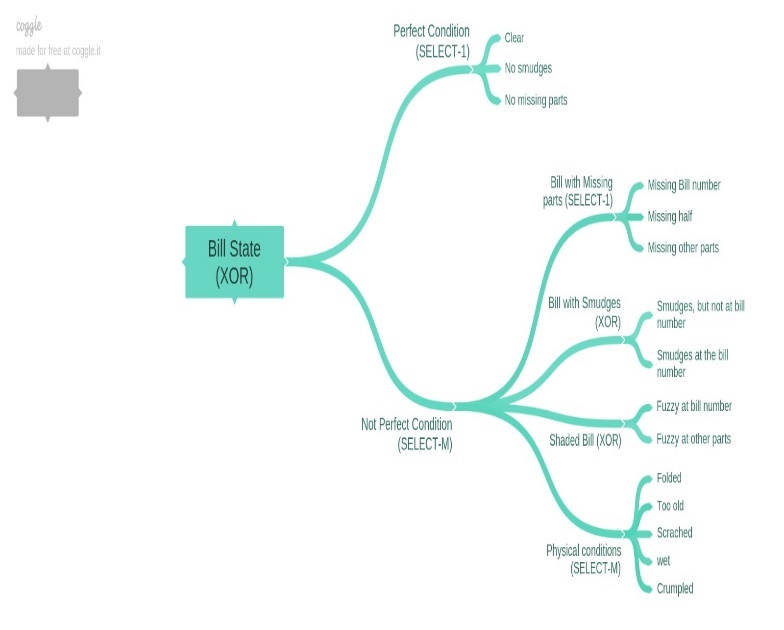


**Figure 5: AI context spanning tree.**

1. **AI Function Input Classifications**

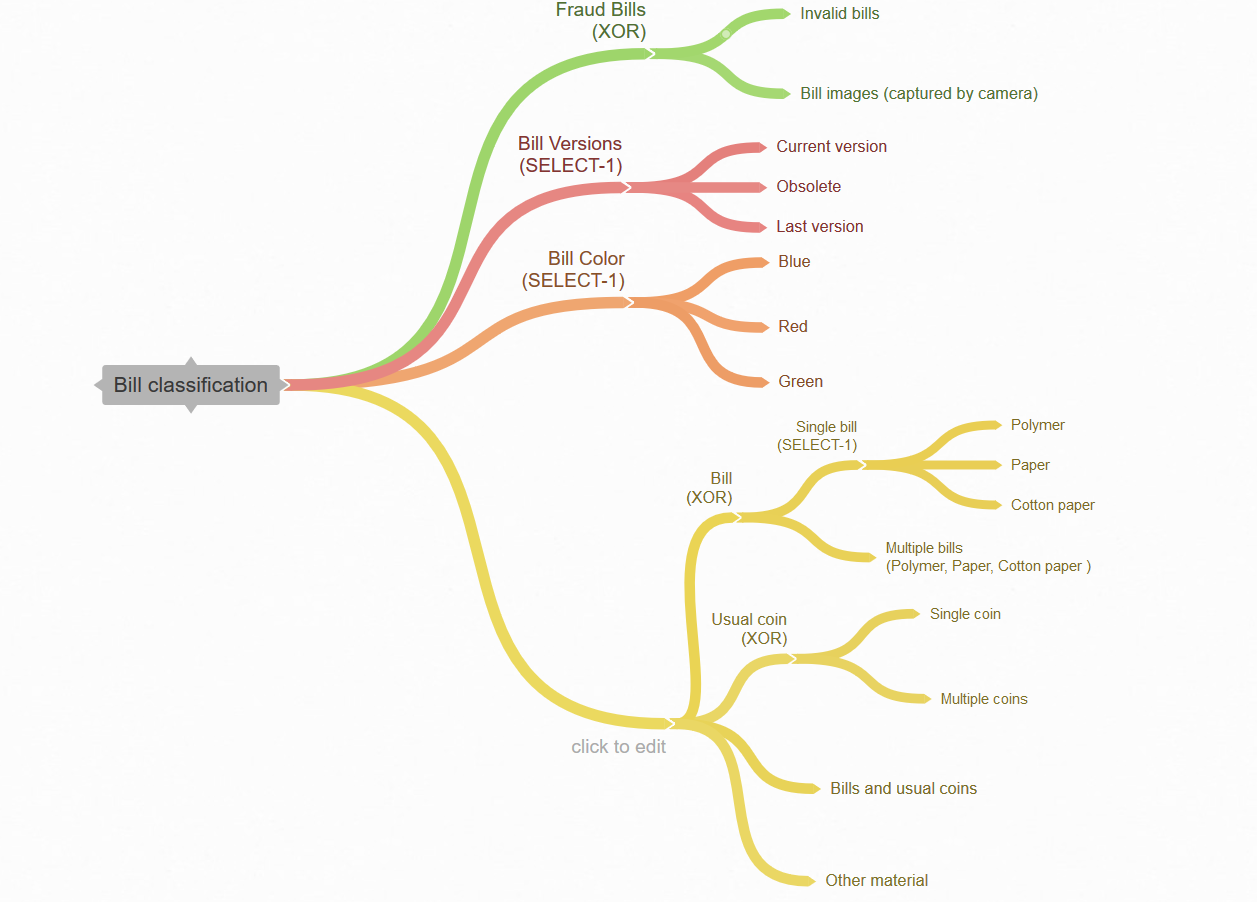
We separated the input data into 2 domains: bill state and bill classification.

**Bill state classifications**



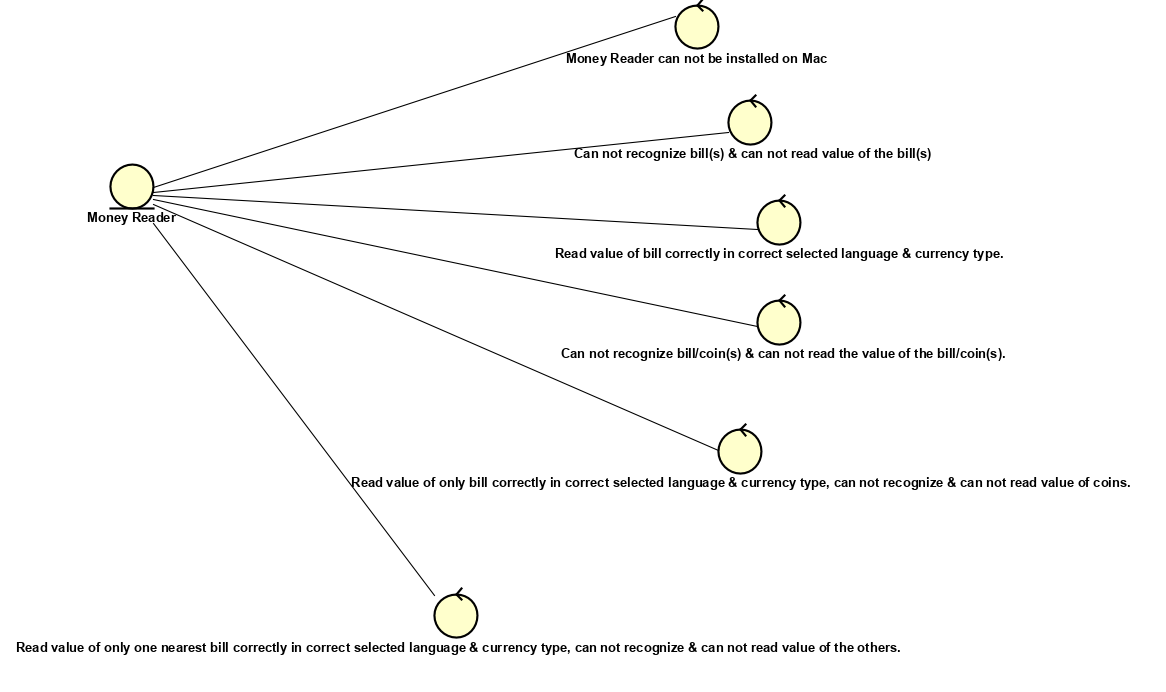
**Figure 6: Bill State spanning tree.**

**Bill classification inputting**



1. **AI Function Output Classifications**

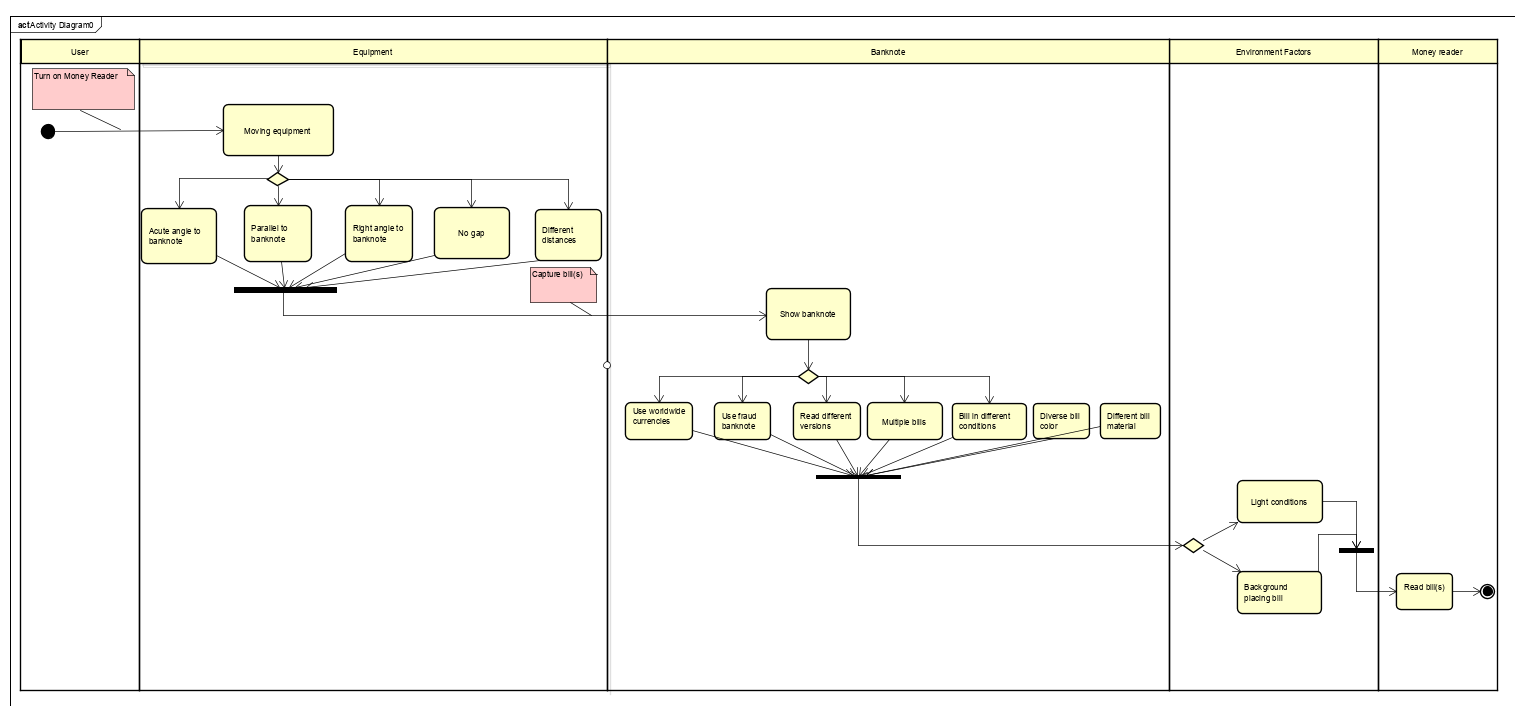
Base on the diverse inputs of currency that defined above and executed under different context, we classified 6 outtput results: a) the money reader can read the value of the bill correctly in correct selected language and currency type; b) the money reader cannot recognize the bill(s) and cannot read the value of the bill(s); c) the money reader cannot recognize the bill/coin(s) and cannot read the value of the bill/coin(s); d) Money Reader can read the value of only one nearest bill correctly in correct selected language and currency type, cannot recognize and cannot read the value of the others; e) money reader can read the value of only the bill correctly in correct selected language and currency type, cannot recognize and cannot read the value of the coins or f) it cannot be installed on the machine (even it is declared as can work well on Mac laptop).



**Figure 8: AI output classifications.**

1. **AI Function Event/Action Classifications**

Figure 7 represents activity diagram of the money reading process with some main steps.



**Figure 9: AI function events.**

* 1. **AI Function Classification Decision Table**

With all classified spanning trees of AI context, AI inputs and outputs above, we built decision tables foreach classification that are prerequisites for building testcases.

1. **AI Context Decision Table**

Table 5 shows random combination of all options that defined in Figure 3, in which “T” means: “True” and “F” means ”False”.

**Figure 7: Bill classification spanning tree.**

## 

## 

## Table 2: AI context decision table.

## Here, the total number of rules: 2^17 (2^number of conditions, in which we excluded 2 conditions of Dell laptop and Mac laptop because money reader application cannot be installed on these two machines).

1. **Bill State Decision Table**

Table 6 shows random combination of all options that defined in Figure 4.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bill State Decision Table** | | |  |  | Rules | | | | | | | | | | | | | | | | | | | | |
| Conditions | Perfect condition | Clear |  |  | T | T | T | T | T | T | T | F | F | F | F | T | F | F | F | F | F | F | F | F | T |
| No smudges |  |  | T | T | T | T | T | F | F | T | T | T | F | F | T | T | T | F | T | T | T | T | T |
| No missing parts |  |  | T | T | F | F | F | T | T | T | F | F | T | T | T | T | T | T | T | T | T | T | F |
|  | Have missing parts | Missing bill denomination |  | - | - | T | F | F | - | - | - | T | - | - | - | - | - | - | - | - | - | - | - | - |
| Not perfect condition | Missing half |  | - | - | - | T | F | - | - | - | - | T | - | - | - | - | - | - | - | - | - | - | - |
|  | Missing other parts |  | - | - | - | - | T | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | T |
|  | Have smudges | Add other shapes but not on bill denomination |  | - | - | - | - | - | T | - | T | F | F | T | - | - | F | F | F | F | T | T | T | - |
|  | Add other shapes at the bill denomination |  | - | - | - | - | - | - | T | - | T | T | F | - | - | T | T | T | T | - | - | - | - |
|  | Not clear | Fuzzy at the bill denomination |  | - | - | - | - | - | - | - | T | T | F | F | T | - | T | F | F | F | T | T | T | - |
|  |  | Fuzzy at the other parts |  | - | - | - | - | - | - | - | - | - | T | T | - | - | - | T | T | T | - | - | - | - |
|  | Other conditions | Folded |  | - | - | - | - | - | - | - | - | F | F | F | - | F | T | T | T | F | F | T | F | F |
|  |  | Too old |  | - | - | - | - | - | - | - | - | - | T | F | - | F | T | F | F | F | T | F | F | F |
|  |  | Scratched |  | - | - | - | - | - | - | - | - | - | - | T | - | T | T | T | F | T | F | T | F | F |
|  |  | Wet |  | - | - | - | - | - | - | - | - | - | - | - | T | T | T | T | T | F | T | F | F | F |
|  |  | Crumpled |  | - | - | - | - | - | - | - | - | - | - | - | - | T | F | F | F | T | F | F | F | F |
| Actions | Money Reader can read the value of the bill correctly in correct selected language and currency type. | |  |  | T | T | T | T | T | T | T | T | T | T | T | T | F | T | T | T | F | T | T | T | T |
| Money Reader cannot recognize the bill and cannot read the value of the bill | |  |  | - | - | - | - | - | - | - | - | - | - | - | - | T | - | - | - | T | - | - | - | - |

**Table 3: Bill state decision table.**

## We have 7 conditions (count all rows at the third column) then total number of rules: 2^7 = 128 rules.

1. **Bill Classification Decision Table**

Table 7 represents random combination of all branches that classified in Figure 5.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bill classification Decision Table** | | | Rules | | | | | | | | | | | | | | | | | | | | | | | |
| Conditions | Fraud bills | Invalid banknotes |  |  | F | F | T | F | F | F | F | F | F | F | F | F | F | F | F | F | F | F | F | F | F |
| Bill images captured by camera |  |  | - | - | - | T | F | F | F | F | F | F | F | F | F | F | F | F | F | F | F | F | F |
|  | Current version |  |  | - | - | - | - | T | T | T | F | F | F | F | T | T | T | T | - | T | T | T | T | T |
| Banknote version | Obsolete |  |  | - | - | - | - | - | - | - | T | T | F | F | - | - | - | - | - | - | - | - | - | - |
|  | Last version |  |  | - | - | - | - | - | - | - | - | - | T | T | - | - | - | - | T | - | - | - | - | - |
| Bill color | Blue |  |  | F | F | T | F | T | - | T | - | - | - | - | - | T | T | F | - | F | F | F | F | F |
| Red |  |  | T | T | - | T | - | T | - | T | - | T | - | T | - | - | T | - | T | T | F | F | F |
| Green |  |  | - | - | - | - | - | - | - | - | T | - | - | - | - | - | - | T | - | - | T | - | F |
| Currency type | Bill | Single bill | Polymer | T | T | F | T | F | F | F | - | F | F | F | - | - | - | - | - | F | F | T | F | F |
| Paper | - | - | T | - | - | T | - | - | T | T | F | T | - | T | T | T | F | T | - | - | - |
| Cotton paper | - | - | - | - | T | - | - | T | - | - | F | - | - | - | - | - | T | - | - | - | - |
| Multiple bills |  | - | - | - | - | - | - | T | - | - | - | - | - | T | - | - | - | - | - | - | - | - |
| Usual Coin | Single coin |  | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | T | - |
| Multiple coins |  | - | - | - | - | - | - | - | - | - | T | T | - | - | - | - | - | - | - | - | - | - |
| Bill and usual coins |  |  | - | - | - | - | - | - | - | - | T | T | - | - | - | - | - | - | - | - | - | - | - |
| Other material |  |  | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | T |
| Actions | Money Reader can read the value of the bill correctly in correct selected language and currency type. | |  |  | T | T | F | F | T | T | F | F | F | F | F | T | F | T | T | T | T | T | T | F | F |
| Money Reader cannot recognize the bill/coin(s) and cannot read the value of the bill/coin(s). | |  |  | - | - | T | T | - | - | F | T | T | F | T | - | F | - | - | - | - | - | - | T | T |
| Money Reader can read the value of only one nearest bill correctly in correct selected language and currency type, cannot recognize and cannot read the value of the others. | |  |  | - | - | - | - | - | - | T | - | - | F | - | - | T | - | - | - | - | - | - | - | - |
| Money Reader can read the value of only the bill correctly in correct selected language and currency type, cannot recognize and cannot read the value of the coins. | |  |  | - | - | - | - | - | - | - | - | - | T | - | - | - | - | - | - | - | - | - | - | - |

## Table 4: Bill classification decision table.

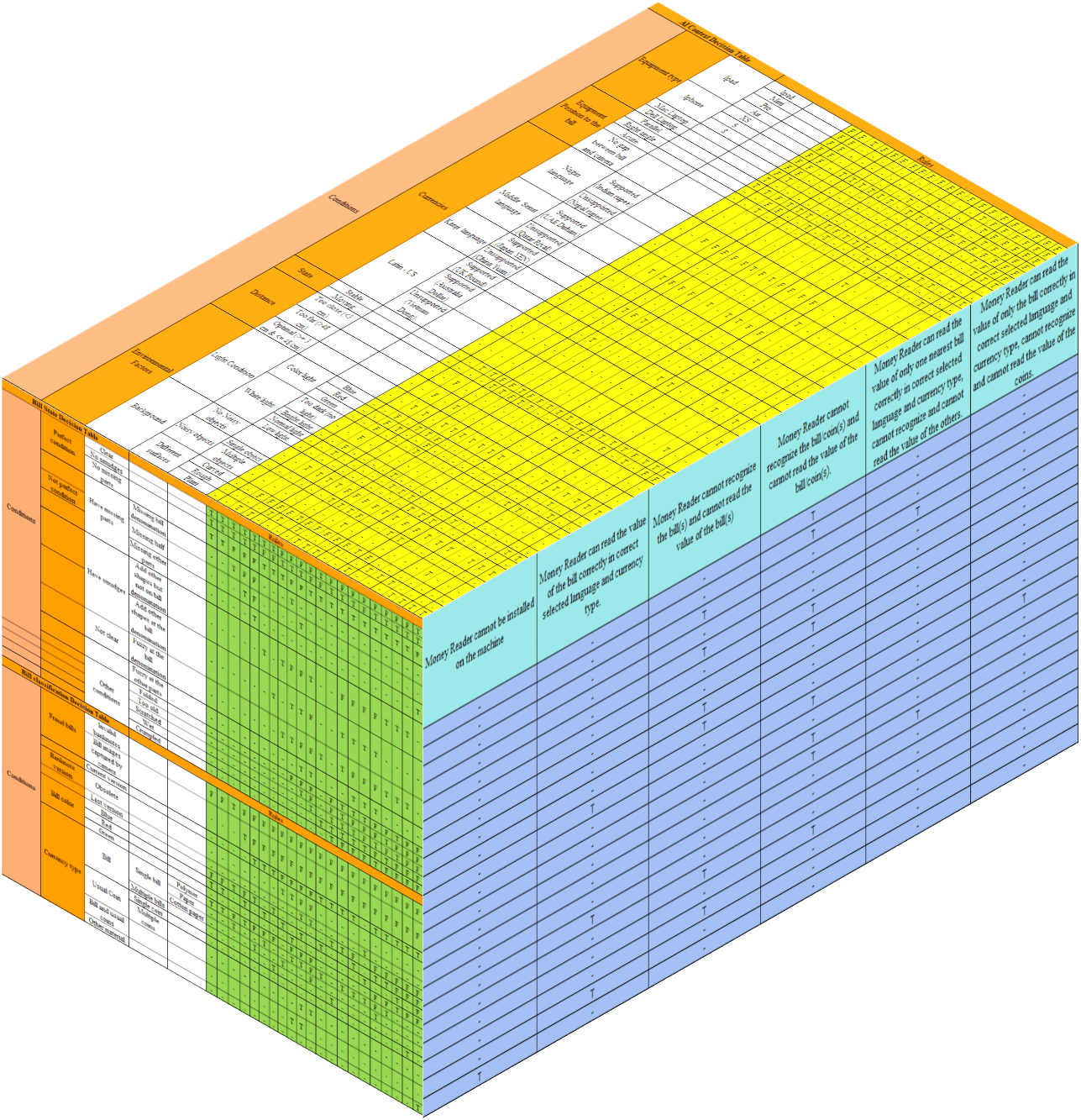
## Total number of rules: 2^12 = 4096 rules.

1. **3D – Decision table**

The concept of 3D-decision table is built on input and context decision tables that shown at above sections.

On the X-axis, we place the decision table of all inputs (bill states merged with bill classifications), on the Z-axis we place context decision table and finally, on the Y-axis we place all out comes that can be occurred base on combination of each input and context rule. That means each context rule can go with each input rule and result a consequence.

For example: Rule 1 in context decision table (on Z-axis) can go with rule 1 in input decision table (on X-axis) to derive the last output on the blue table on Y-axis and so on.



**Figure 10: 3D-decision table.**

* 1. **Test result**

Figure 11 below represents the total number of AI testcases that we implemented goes with number of passed/failed testcases and defects found in each business checking.

Figure 12 shows total number of testcases designed using decision table method and scenario test method in each business checking correspondingly.

With the definition of 3D decision table above, we built 21 testcases base on the combination of the first random 21 rules of input and context decision tables. Moreover, we executed some more testcases that are combination of other context and inputs options, excluding above 21 rules.

**Figure 11: AI test summary.**

**Figure 12: Summary of AI testcases and test methods.**

**5 TEST COMPLEXITY COMPARISON**

To get general view of our testing result, we do a comparison between conventional and AI testing in the figures below. Moreover, this comparison also gives us the advantage of AI testing in uncovering defects.

Figure 13 consists of total number of testcases along with total number of passed/failed testcases and total number of defects uncovered in conventional and AI testing.

Besides that, we collect total number of testcases that created in each test model using three major test methods and represent them in the figure 14.

**Figure 13: Test complexity comparison between conventional testing and AI testing.**

**Figure 14: Comparison of testcase number designed in each test method.**

**6 BUG COMPARISON**

In this experiment, some defects cannot be found in conventional testing but can be uncovered in AI testing model and vice versa. This result is achieved by using 3D-decision table technique that supports the definition of wide range of testcases. In Figure 16 below we can see 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.

Moreover, in Figure 15, we show total number of defects classified into 4 categories of bug such as: device compatibility, function error, design error and missing feature.

**Figure 15: Bug classification comparison between conventional and AI testing. Figure 16: Capacity of bug cross-detection between conventional and AI testing.**

**7 AI FUNCTIONS TESTING QUALITY ASSESSMENT AND**

**CONVENTIONAL TESTING ASSESSMENT**

|  |  |
| --- | --- |
| **Conventional Testing**  General pass/fail percentage    **Figure 17: Conventional general test result.** | **AI testing**  General pass/fail percentage    **Figure 20: AI general test result.** |
| Testing quality assurance results with metrics and assessments  **Figure 18: Conventional testing quality assurance results with metrics and assessments.** | Testing quality assurance results with metrics and assessments  **Figure 21: AI testing quality assurance results with metrics and assessments.** |
| Test result by category **Figure 19: Conventional test result by category.** | Test result by each group of business checking  **Figure 22: AI test result by business checking.** |

**8 CONCLUSIONS**

In this present work we focused on the challenges of using conventional software testing methods to test AI applications. Our work concludes that AI functions cannot be adequately tested with conventional methods and require different strategy such as classification-based to completely validate the AI application. We tested the currency denomination identifying AI application LookTel money reader application using conventional approaches by decision table, scenario and equivalence partition methods and comparing with AI function testing by classification-based and rule-based approaches. We used AI function test modeled from context, AI function input and AI function output modeling and 3D decision table to generate the test cases. We evaluated the quality assurance of AI function using metrics 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 require different approach and in future studies should focus on validating the AI application using AI-based approaches so that the entire flow is automated.

**9 REFERENCES**

1. https://www.tractica.com/research/artificial-intelligence-market-forecasts/

2. Gao, Jerry &amp; Tao, Chuanqi &amp; Jie, Dou &amp; Lu, Shengqiang. (2019). Invited Paper: What is AI Software Testing? and Why. 27-

2709. 10.1109/SOSE.2019.00015.

3. A.M. Turning, “Computing Machinery and Intelligence,” Mind, vol. 59, pp. 433-460, 1950.

4. Saygin A P, Cicekli I, Akman V. Turing Test: 50 Years Later[J]. Minds &amp; Machines, 2000, 10(4):463-518.

5. Li L, Lin Y L, Zheng N N, et al. 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. Intelligence Testing for Autonomous Vehicles: A New Approach [J]. IEEE Transactions on

Intelligent Vehicles, 2017, PP(99):1-1.

8. Zhang N, Wang F Y, Zhu F, et al. DynaCAS: Computational Experiments and Decision Support for ITS[J]. IEEE Intelligent

Systems, 2008, 23(6):19-23.

9. Wang F Y. Parallel control and management for intelligent transportation systems: concepts, architectures, and applications[J].

IEEE Transactions on Intelligent Transportation Systems, 2010, 11(3):630-638.

10. Li L, Wen D. Parallel Systems for Traffic Control: A Rethinking[J]. IEEE Transactions on Intelligent Transportation Systems,

2016, 17(4):1179-1182.

11. Li L I, Lun L Y, Pu C D, et al. Parallel Learning — A New Framework for Machine Learning[J]. Acta Automatica Sinica, 2017.

12. Li L, Lin Y, Zheng N, et al. Parallel Learning: a Perspective and a Framework[J]. IEEE/CAA Journal of Automatica Sinica,

2017, 4(3):389-395.

13. Straub, Jeremy, and Justin Huber. &quot;A characterization of the utility of using artificial intelligence to test two artificial intelligence

systems.&quot; Computers 2.2 (2013): 67-87

14. Website: http://www.looktel.com/moneyreader

15. Apple store page: https://itunes.apple.com/us/app/looktel-money-reader/id417476558?mt=8