A SUPPORT VECTOR MACHINE BASED APPROACH FOR IMPROVING ACCURACY AND PERFORMANCE OF TEST ORACLES

MUHAMMAD ELRASHID YOUSIF MOHAMED

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITI MALAYA
KUALA LUMPUR

20217

A SUPPORT VECTOR MACHINE BASED APPROACH FOR IMPROVING ACCURACY AND PERFORMANCE OF TEST ORACLES

MUHAMMAD ELRASHID YOUSIF MOHAMED

DISSERTATION SUBMITTED IN PARTIAL

FULFILMENT OF THE REQUIREMENT FOR

THE DEGREE OF MASTER OF SOFTWARE ENGINEERING

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY MALAYA
KUALA LUMPUR

2017

UNIVERSITY OF MALAYA
ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: Muhammad Elrashid Yousif

Matric No: WGC130002

Name of Degree: Master of Software Engineering

Title of Dissertation: A Support Vector Machine Based Approach for Improving Accuracy and Performance of Test Oracles.

Field of Study: Formal Methods

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date: 27th November 2017

Subscribed and solemnly declared before,

Witness's Signature Date: 27th November 2017

Name:

Designation:

Abstract

One of the key structures in software development is software testing, where there is an increasing need to deal with the issue to provide automated test oracles. Test oracles are simplified and reliable sources that guide testers to undertake a testing process and evaluate faults detected in software. Throughout the years, there has been countless of research conducted on different formats of test oracles, all withholding a similar objective to conclude on whether the test oracle chosen an improvement to software is testing. The objective is to the obtain the challenges currently with existing test oracles and with that identify an approach to address those challenges, using methods such as black-box testing and applying pattern recognizers such just support vector machines as an automated test oracle. In this research, Support Vector Machine, a pattern recognizer based on automated test oracle is introduced to handle mapping and comparison automatically. Previous test oracles, such as Info Fuzzy Networks (IFN) introduced on regression testing contain numerous drawbacks including its limitations to only a single form of testing and its inability to acquire data from other test oracles. Artificial Neural Networks (ANN) based on a single network is a test oracle introduced, in favor of IFN, as it generated accurate data of 91.83% when undertaking testing process. Nevertheless, single network oracle based on ANN has a central drawback and that is its inability to test complex software, making it unreliable. The pattern recognizer SVM is functioned to improve classification performances which will be applied for detecting faults in a testing process. It also contains the capability of solving functional estimated problems and pattern recognition as well as able to undertake complex software. The implication of this research is to provide to software tester different testing methods to apply to projects that may improve accuracy rate as well as reduce cost.

Abstrak

Salah satu struktur utama dalam pembangunan perisian adalah pengujian perisian, di mana terdapat peningkatan yang perlu untuk menangani ujian oracle automatik. Ujian Oracle adalah sumber yang mudah dan boleh dipercayai untuk membimbing penguji menjalankan proses ujian dan menilai kesalahan yang dikesan dalam perisian. Selama bertahun-tahun, banyak penyelidikan telah dijalankan dalam pelbagai bentuk ujian Oracle untuk untuk menyimpulkan sama ada ujian oracle yang dipilih meningkatkan pengujian perisian. Objektif ujian-ujian ini adalah untuk memahami cabaran-cabaran yang sedia ada dan dengan itu mengenal pasti pendekatan terbaik untuk menangani cabaran-cabaran tersebut. Antara kaedah-kaedah yang digunakan adalah pengujian kotak hitam dan pengenal corak seperti mesin vektor sokongan sebagai ujian oracle automatik. Dalam kajian ini, Mesin Vektor Sokongan, sebuah pengenal corak berdasarkan ujian oracle automatik diperkenalkan untuk mengendalikan pemetaan dan perbandingan secara automatik. Merujuk kepada ujian oracle sebelumnya, seperti Rangkaian Info Fuzzy (IFN) yang diperkenalkan pada ujian regresi menunjukkan pelbagai kelemahan termasuk batasan untuk hanya satu bentuk ujian dan ketidakupayaannya untuk memperoleh data dari ujian oracle lain. Rangkaian Neural Buatan (ANN) berdasarkan rangkaian tunggal merupakan ujian oracle yang memihak kepada IFN diperkenalkan kerana ia menghasilkan data yang tepat sehingga 91.83% apabila menjalankan proses ujian. Walau bagaimanapun, ketidakupayaan oracle rangkaian tunggal berdasarkan ANN untuk menguji perisian kompleks merupakan kelemahan utama dan menjadikan ANN tidak boleh dipercayai sepenuhnya. Pengenal corak SVM berfungsi untuk meningkatkan prestasi klasifikasi untuk mengesan kesalahan dalam proses ujian. Ia juga mengandungi keupayaan menyelesaikan masalah fungsian yang dianggar dan pengecaman corak serta dapat menguji perisian yang kompleks. Implikasi kajian ini adalah untuk memberikan

penguji perisian kaedah ujian yang berbeza untuk meningkatkan kadar ketepatan serta mengurangkan kos dalam projek yang dilaksanakan.

Acknowledgement

This dissertation could not have been possible without the help, guidance and support from many people. I would like to sincerely thank my supervisor, Dr. Mumtaz Begum, who has been there to promptly guide to the completion of this research. I was fortunate enough to work with Dr. Mumtaz and learn from her experience and dedication, and for that I am grateful, so thank you very much Dr. Mumtaz.

I would like to also thank the Faculty of Computer Science and Information Technology in the University of Malaya for facilitating the completion of this work. And lastly, I would to thank my family, my father Dr. Elrashid Yousif who is the main influence for me to push myself to obtain a higher education. My mother Ikhlas Alamin for her love and support and my siblings Musab, Yousof and Aminah for their encouragement and support, and making this possible. It could not have been done without your help.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviations	Description
SUT	System Under Test
ANN	Artificial Neural Networks
IFN	Info Fuzzy Networks
HMM	Hidden Markov Model
SVM	Support Vector Machine
I/O	Input/output
FSVM	Fuzzy Theory Support Vector Machine
LibSVM	Library of Support Vector Machine
KNN	K-Nearest Neighbor
SDLC	System Development Life Cycle

Chapter 1

Introduction

This chapter provides an introduction to the research inclusive of a number of relative aspects. First, it describes the overview of the research and the background of the problem, which is then followed by the objectives and the methodology of the research. Next, it describes the scope of the research, and finally the significance of the research.

1.1 Overview

Software testing is the process of evaluation conducted on the quality and reliability of a software product. It is a crucial part in the software development life cycle. Software testing has been applied to generate results to indicate software faults, these faults are conveyed with a variety of software testing methods, used accordingly to their automated tools. Whether the results of these testing processes have succeeded or failed, a foremost concept is firstly highlighted on how the test underwent.

In other words, software testing without automated tools is complicated and difficult to complete. Therefore, automated approaches are recommended as they provide results with better reliability, cost saving and less time consuming. There are a large variety of software testing approaches in the field of software testing. Levels of testing such as *unit, integration, system and acceptance testing* as well as test methods such as *performance stress and regression testing* are considered with different testing paradigms such as *White-Box and Black-Box* approaches. Each of these methods contains a specific process in identifying software faults. For example the white-box (Open-box) test

represents the module structure while the black-box (Closed-box) test approaches are focused with the functionality of the modules (Agarwal, Tamir, Last, & Kandel, 2012).

To develop quality software or applications, a proper testing method is required. However, as software testing has become an expensive process, time consuming and depending on the approach being applied it could be unreliable, the testing phase must balance these factors. Whittaker (2000) explained the difficulties of software testing and how to approach a solution towards those difficulties, by inheriting a clearer view of the testing problems. Software testing was divided into four phases:

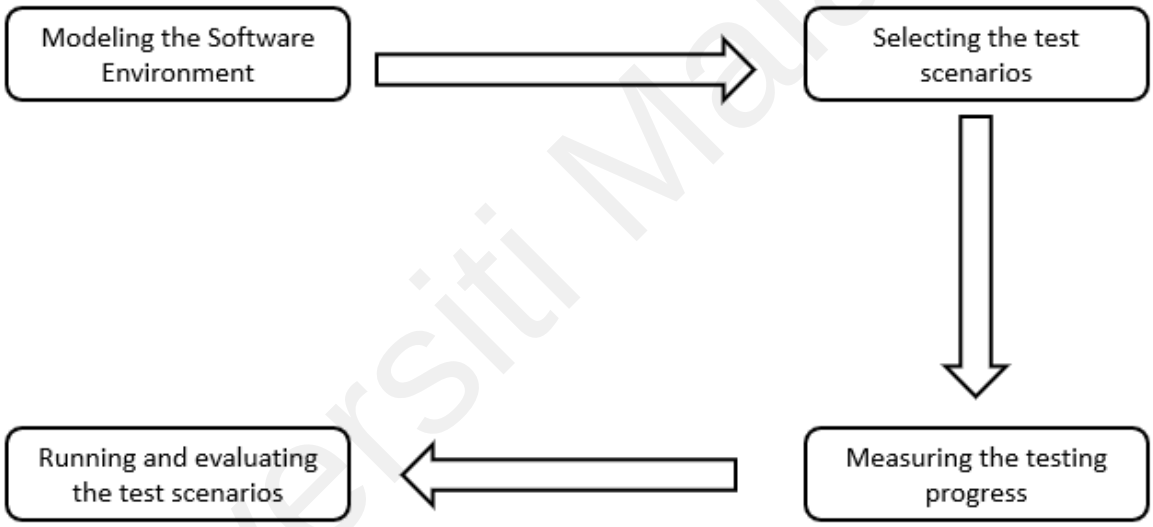

Figure 1.1 Software Testing Phases (Whittaker, 2000)

In the running and evaluating test scenario phase, testers will evaluate whether the test scenarios contain any faults such as manipulated/faulty data. (Shahamiri, Kadir, Ibrahim, & Hashim, 2011). After these phases are executed and generated, a fault report is constructed based on the output (Whittaker, 2000). By applying these four phases, testers will be able to conclude the quality of the tested software. Many software companies tend to stress enough

attention to the testing process and it's probably due to the cost and timeline taken to conduct a software test, hence the product becomes unreliable with a higher risk of failure.

Thus, it is important to conduct research on differentiation of software testing procedures to continue in order to find suitable approaches that will decrease the testing cost and simultaneously increase the efficiency of the testing process. *Automated testing* is one of the approaches applied for software testing. Previous researches have shown different types of automated testing approaches that have improved software testing methods in comparison to manual testing by using intelligent approaches, algorithms and linear equations.

1.1.1 Test Oracles

Test oracle is a typing software testing method with a logic that determines if an application is executed appropriately. In a layman's term, it would have three capabilities executing as follows:

i. A generator: to provide expected results for each test.
ii. A comparator: to compare between the expected and obtained results.
iii. An evaluator: to determine whether the comparative results are sufficient.

These three capabilities are applied in the running and evaluation test scenarios. A test oracle is applied to verify test case results executed by System under Test (SUT). This is used as an important source of information on how the SUT should behave and a verification to actual outputs correctness (Shahamiri et al., 2011). In a cross-referencing understanding, the SUT outputs required to be verified is known as *actual outputs*, and these actual outputs that are evaluated by the correct results known as *expected outputs*.

Normally, verifier compares the actual output and the expected outputs to conclude the validity of the results.

Automated approaches must provide the appropriate expected outputs for any input combinations and to do so, it depends on the software that is being tested, and the right software testing paradigm applied. There are countless of software testing paradigm, however in the recent findings for software testing, two approaches standout to be suitable format for software testing processes while the rest have been disregarded. These two software paradigms are *white-box* and *black-box approach*.

In addition, to the black-box approach is found to be favorable by testers due to its focus between input and outputs, without peering into the cause of these relations (Whittaker, 2000). It is not necessary for a software testing paradigm to function with a pattern recognizer. Approaches such as the black-box approach have been used by human oracles (also known as the traditional method). However due to the complexity of the current software system and the increase in number and size of the programs being tested, testers are unable to rely on human oracles and thus the approach becomes obsolete. Thus, with automated test oracles and obtainment of a pattern recognizer, these approaches are able to sustain their abilities as a software testing paradigm.

In software testing, different tools will be used to undertake the test and generate end results. There are numerous types of testing in different platforms and normally testers will have to obtain the right automation tool and test approaches accordingly to the requirement of the software project. These automation tools are known as *pattern recognizers*. These pattern recognizers are Artificial Neural Networks (ANN), Info-Fuzzy Networks (IFN),

Hidden Markov Model (HMM), Generic Algorithm (GA), Support Vector Machine (SVM) and many more.

1.2 Research Motivation

In this stage of software development, software testing is known to be the most important section of software development life cycle. When developing software, testing has to be extremely selective, detailed and applied numerous times and this is where researchers and testers have intervened to obtain a solution on making software testing processes more reliable, less time consuming and less costly. There are a large number of researches on automated test oracles and improvements on software testing and most of these researches have used different objectives, steps and tools to obtain these improvements. In addition, existing researches contain both successive and failed experimentation. One of the latest automated oracles introduced was multi-network oracles applied as an automated test oracle which has rendered a good improvement in software testing concerning neural networks. Therefore, the main objectives of software testing researchers are to conclude and innovate an improvement for software testing. In this research, a similar concept is proposed but with a different objective and tool which is Support vector machine impacting on automated test oracles. Why a research on this domain is simply because Support Vector Machine (SVM) is an innovative automated test approach that has the ability to conduct a software testing. Previous researches conducted on SVM concluded that it's a reliable algorithm (Cortes & Vapnik, 1995; Chih-Wei Hsu, Chih-Chung Chang, 2008;Xue, Chen, & Yang, 2011) and that's what this research is about. To evaluate and conduct what effect and results this tool can provide when applied as an automated test oracle in the categories of reliability, less time consuming and less costly.

1.3 Problem Statement

Being an important phase, software testing can be redeemed unreliable, time consuming and costly. This is depending on the form of testing being undertaken. Testing method such as human oracle has concluded to be extremely unreliable due to the impossibility of completion. Nevertheless, testers began implementing automated approaches to improve the methods of software testing. However, even if the method is an automated approach, it may still be rendered as low quality, unreliable, time consumption and increase of cost (Shahamiri, Wan-Kadir, Ibrahim, & Hashim, 2012), this is mainly depending on the method that has been used to run the software testing process. However, this being a research and before stating facts or experimentations to be undertaken, questions need to be asked.

Previously, researches have been conducted on automated testing, implementing software testing algorithm using pattern recognizers such as *Info-Fuzzy Networks (IFN)* and *Artificial Neural Network (ANN)*. Results on these pattern recognizers have shown improvements as compared to other software testing methods. Shahamiri et al. (2011), deduced that by using suitable testing method and pattern recognizers, the practicality and accuracy obtained a dramatic increase and the error rate deteriorated, moreover it is also explained that both recognizers show limited improvements to specific criteria.

Applying *Info-Fuzzy Networks (IFN)* based oracles may be suitable for knowledge discovery and data mining methodology. IFN joins together the process of selected features and classifications, which is crucial in decision making process. This is by identifying the input vector in the software testing. With IFN, there is an interaction between the inputs and outputs that are represented by the data provided; this is all due to the multilayered decision graph in the IFN.

IFN when applied to an automated oracle, is identified in two phases (Agarwal et al., 2012), which are the *IFN learning phase* and *Evaluation phase*. In IFN learning phase, IFN is trained to learn and is prepared with the format of architecture and formulas before being presented to the system. Therefore, all IFN parameters are solely determined by the learning procedure. On the other hand, similar to ANN, the IFN evaluation phase is much simpler. This is due to applying the results from the records being tested on the ANN to train the IFN. Hence, records would be tested on the IFN to generate the result. The next procedure would be to compare the results and to conclude whether the actual outputs are valid. The problem with this IFN method applying on the automated oracle is that the IFN has to be trained first opposed to the ANN where the parameters are at the network of the software architecture (Shahamiri et al., 2011) Hence applying IFN may results to time consuming and less efficiency. As a result, accuracy and time are the main factoring issues in automated test oracles.

1.4 Research Questions

The general question this research tries to answer is:

How to develop an automated test framework that will address the challenges automatically with adequate quality and cost?

More questions following the research objectives:

Research Objective 1:

RQ1.What are the challenges with existing software test oracles?

Research Objective 2

RQ2. Why support vector machine?

RQ3. What are the advantages and disadvantages of applying support vector machine as a pattern recognizer?

RQ4. How does support vector machine address the challenges faced by existing pattern recognizers?

Research Object 3

RQ5. How is the effectiveness of the approach being measured?

RQ6. What is the case study being used?

RQ7. Against what test oracles is the approach compared with?

1.5 Objectives

This research aims to obtain a research on automated approaches for software test oracles and how the automated approaches address challenges and compare with one another. The objective of this research is as follows:

1. To ascertain the challenges in the existing software test oracle.
2. To identify a pattern recognition approach to address the challenges in an automated oracle in software testing.
3. To measure the accuracy and the performance (in term of processing time) of the approach by the application with a selected case study and the comparison with other existing pattern recognizers.

1.6 Research Scopes

A constructed research to identify an automated software test oracle using pattern recognizers on a pattern recognition approach. The pattern recognizer would assume a black-box test approach. The experiment would be constructed in a format of case studies focusing on the courier domain using specific data from the courier industry to execute the testing process.

1.7 Research Contribution

Software testing is an ideal phase for testers and software developers. It is the single phase that deduces for these roles whether the software they have constructed is qualified, quality and accurate software achieving the necessary standards.

Testers require an approach that could improve the testing process. An approach that can provide accuracy and quality without consuming so much time and cost. This research contributes an automated software testing approach that can detect faults/bugs with higher accuracy and more importantly solve the issues of the existing oracles.

1.8 Significant of the Research

The importance of this research guides software testers to understand the purpose, process and requirements for automated solutions when it comes to software testing. This research will be able to show testers the difference in types of testing methods and how it could collaborate with their currents projects. Also, manual testing has become costly and depending on the size of a project, it can be extremely time consuming. The use of human testing increases the human error, causing more software project requirements to move to

an automated approach. Therefore, the research exploits the problem and proposes solutions to software testers on the benefits and limitations of different testing methods.

1.9 Dissertation Outline

This dissertation covers some discussions on specific issues associated to software test oracle specially to support oracle automation. It also describes the new proposed automated oracle framework in details. The dissertation is organized as follows:

Chapter 2: It discusses the findings of the literature review on software testing and test oracles. First, in this chapter, software testing activities are explained and some existing solutions to automate each activity are presented. Next, software paradigms used as the basis for a software testing method is explained, including the comparisons between different paradigms, there drawbacks and limitations. Next, a description on pattern recognizers, there importance and the role they play in a software testing method. Next, a relative and existing test oracle described in detail based on ANNs and IFNs, the success rate of these existing oracles. Moreover, a comparative study among automated oracle models is presented as well. Finally, Support Vector Machine, its role in this method for this significant research, its drawbacks and limitations and how it will be applied.

Chapter 3: This chapter describes the proposed research approach design and procedure that is being applied. It also describes the design and finally it explains the experimentation process and evaluation criteria that are considered in this research.

Chapter 4: It explains the proposed model in details. The discussion includes the steps taken in the experimentation process in detail. The results and statistics that the proposed model has processed based on the two selected case studies. In addition, this chapter describes the evaluation model that is used to assess the proposed approach followed by data analysis.

Chapter 5: This chapter explains the evaluation phase in details corresponding to the results of the experimentation design and implementation of the automated tool. A comparative study among oracle models mentioned in chapter two and the proposed one is presented.

Chapter 6: It explains the conclusion based on the results evaluated, statements on the researches achievements and contributions. It is followed by the research summary and suggestions for future work.

Chapter 2

2 Literature Review

In this chapter, background on software testing of how methods have improved software testing, a review on automated test oracles and software test activities is discussed. It begins with the discussion of the conventional methods and its drawbacks, followed with a discussion on the testing activities and solutions to automate them. Next, description on the components required to conduct a testing procedure such as the software testing paradigms and pattern recognizers that have been applied to software testing methods. Finally, a pattern recognition approach on automated oracles is reviewed based on how to manage and handle the challenges such as IFN on regression testing and ANN as a multi-network oracle as well as SVM.

2.1 Background

This section introduces the aspect of software testing, concentrating on black-box approach. Software failures caused by human errors, resulting in program faults, is a deviation of the software from its expected function (Agarwal et al., 2012). It is well known that the conduct of software testing is to conclude the processes of evaluating the quality and reliability of software products. This statement depends on the form of testing that is being conducted. Human testing/manual testing has been a greatly used format in the beginning of software testing. Throughout the years, this method has deteriorated due to the enormous cost and risk failures. Even though faults may be prevented by optimizing the programs used for the testing process, it is complex and rendered in occasions impossible to complete. Software testing being one of the critical stages the software development life

cycle and with the continuation of this form of human testing, investors, testers and owners may achieve software of low rendered quality, highly unreliable form of testing, time consuming and costly. Even in this present day, IT companies are still using this form of manual testing for their applications. However, it is well to note that companies are becoming aware of the disadvantages of existing form of testing and begins to apply the automated approach. With this problem being highlighted, a solution has arisen and that is automated testing which has proved investing in the testing infrastructure can save a tremendous amount of expenses.

2.2 Software Testing Activities

2.2.1 First Activity: Modeling the Software Environment

The first activity is to provide the logical environment model. This is to ensure that the software to be tested is aligned and connected with the necessary components. These different components usually communicate with the software, depending on the framework it could be sending messages or file system. Methods that can simulate the interfaces may be helpful in order to automate the processes.

2.2.2 Second Activity: Selecting Test Scenarios

In this activity, testers must select and construct proper test cases that can cover the core functionalities of the software being tested. Test cases can be very large to generate and executed and hence it's important to filter the core functionalities (Whittaker, 2000). Test case generation methods can either be black-box or white box approaches.

In addition, in this activity, developers select approaches that generate effective test cases such as:

1. Increasing probability to finding errors.
2. Concentration on core functionalities
3. Be the best of its kind.
4. Be neither too complex nor too simple.

2.2.3 Third Activity: Running and Evaluating Test Scenarios

After preparing the test cases, testers run the cases on the SUT, generate the results and evaluate the faults. The SUT provides the *actual outputs* while the *expected outputs* are proved by the test oracle (automated test application). To simulate an automation for this activity, methods are required to map the input domain to the corresponding output domain (Whittaker, 2000). These two domains are constructing the entire environment and will go through an automated comparator that validates the actual output.

2.2.4 Fourth Activity: Measuring Testing Progress

The final activity is to measure the process of testing; this is to generate what is the stature of the test process and when the testing process can be completed. This test process can be indicated based on the number of bugs predicted to be in the software and the probability of those bugs being discovered.

2.3 Software Testing Paradigm

One way to classify a testing procedure is to evaluate test designs and execution by contemplating the availability and details of a module structure (Agarwal et al., 2012). In

other words, which procedure is most suitable to a software paradigm? White-box (Open-box) testing makes complete usage of the module structure, while Black-box (or Closed-box) testing methods are only focused on the functionality of the modules. The main purpose of Black-box is to verify the input data and the generated outputs of the program compiling with the specifications disregarding the actual structure of the program (Agarwal et al., 2012). Hence a comparison between a black-box testing paradigm without an oracle and a black-box testing with an oracle will clearly show the advancement of the automated test oracle.

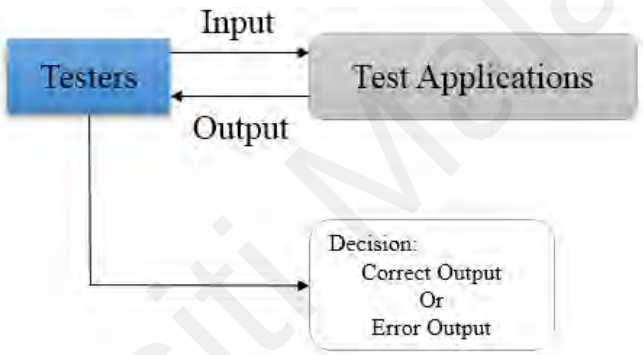

Figure 2.1 Black-Box Testing Paradigm (Agarwal et al., 2012; Shahamiri et al., 2011)

Figure 2.1 shows a typical black-box testing paradigm without an oracle. This is usually how human testing is conducted. This is due to the unavailability of Software under Test (SUT) to produce actual outputs, and compare with the expected outputs by the test applications. The actual output is generated by the testers from their knowledge and interpretations of the program specifications. These "assumed" actual outputs are then compared with the expected output that has been generated from the test applications, leaving the testers to deduce whether the outputs are correct or an error.

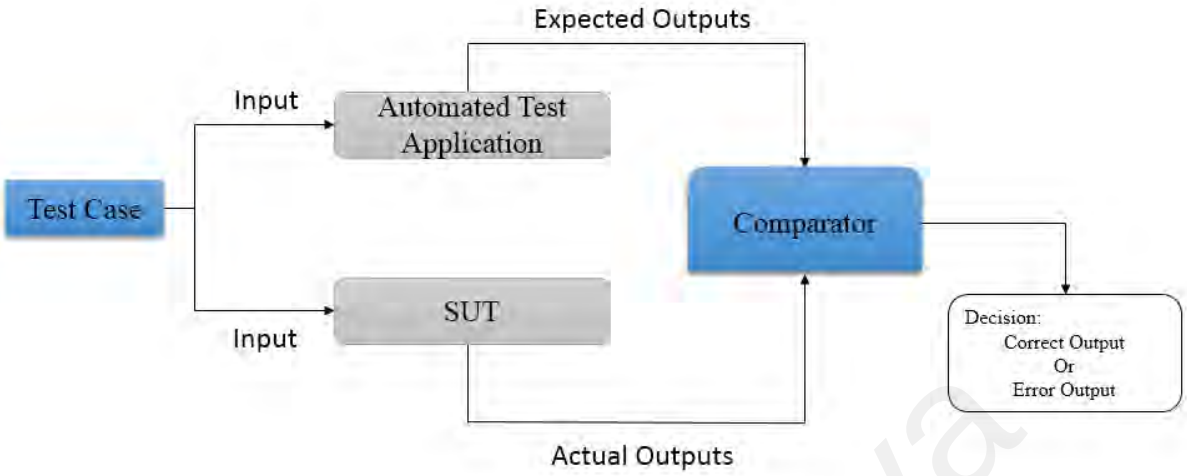

Figure 2.2 Black Box Testing Paradigm with an Oracle

Figure 2.2 is a black-box testing paradigm with an oracle and its functionality has been proved by testers to be more efficient and accurate when compared to Figure 2.2. This is due to the fact that the decision output is not produced by the tester's knowledge rather than it is actually being calculated into an accurate decision. This is credit to the automated oracle in place and the SUT to compare the outputs (Shahamiri, Kadir, & Ibrahim, 2010). As the differences is clearly shown between the two figures, the expected output from the automated test application is not send back to the testers/test cases to produce an actual output. More so, the testers/test case do not have to produce the actual output from their knowledge as that's of SUT's functionality (Shahamiri et al., 2012). Hence the automated test application generates the expected outputs and the SUT generates the actual outputs. Both outputs are then compared to generate a fault report. Conducting manual testing has for a long time contained a drawback of combinatorial explosion of possible test cases, hence the presentation of an automated approach that allows testers to take advantage of this using Input/output relationship programs to generate and to identify the probability of

correctness on relatively small set of test cases. Within this report, testers are able to deduce a decision more accurately whether the output is correct or an error (Schroeder, Faherty, & Korel, 2002).

2.4 Test Oracle (Oracle-based testing)

Test oracles are simplified and reliable sources that guide testers to undertake a testing process and evaluate faults detected in a software. In a detailed manner, test oracles determine how software under test operates and evaluate the actual results produced by the software. Throughout the years there has been countless of research conducted a different formats of test oracles, all withholding a similar objective to conclude on whether the test oracle chosen is an improvement to software testing.

2.4.1 Types of Test Oracles

1. Random Testing – is a black-box test data-generation approach. This test oracles selects test cases to match an operational profile that uses statistical models in order to predict the software under test reliability.
2. Cause-Effect Graphs and Decision Tables –
3. Generic Algorithms (GA) – is a test oracle used for condition coverage testing which address conditional statements. Since manual searching to find test cases that increase condition coverage is difficult and automated test oracle approach such as GA is proposed to generate effective test cases.

4. Info-Fuzzy Networks (IFN) - IFN is adequate on knowledge discovery and data mining methodology and is concerned in the combination of feature selection and classification.
5. Artificial Neural Networks (ANN) - ANN is designed to replicate the structure and information processing of the brain. Instead of neurons, which are components of the brain, ANN's components are computational units. A neural network is from one or more layers that convey the process during input/output testing applications.

2.5 Pattern Recognizers

Software testing can be applied in different formats for different purposes, but in the end its functionality is to generate results for testers. Therefore, the matter has always been on how well an application has been tested. All developers understand the frustration of having software bugs reported by users and the questions are always how the bugs escape the testing procedures. With that note, researchers and testers have always been searching for alternative solutions to reduce the number of faults in software test. The implementation of pattern recognizers could be one of the solutions due to the fact that pattern recognizers can simplifies testing process and a tool that solves ~~a~~ problem of ~~a~~ test case.

Pattern recognizers, are models that function as a machine learning and data mining. In general descriptions, these models are the branch of artificial intelligence focusing on the pattern recognition and regularities in data. There are many types of pattern recognizer such as shown in the table below.

Table 2-1 Types of Pattern Recognizers

Pattern Recognizers	Applied Software test oracles
Artificial Neural Networks (ANN)	• Single-Network Oracle • Multi-Network Oracle
Info-Fuzzy Networks (IFN)	• Regression Testing
Hidden Markov Model (HMM)	
k-Nearest Neighbors (KNN)	
Logistic Regression	
Relevance Vector Machine (RVM)	
Dynamic Time Warping (DTW)	
Generic Algorithm (GA)	• Condition Coverage Testing • Dynamic Test Generator
Redundant Computation	
Support Vector Machine	• Classification Methods • Structural Large Margin Classifier • Hybrid Classifier

2.6 Artificial Neural Networks (ANN)

ANN is one of the most widely studied and used pattern recognizers in software testing and implications on software behavior. ANN is mainly applied as it performed better than some other pattern recognizers. Its benefits are greater than its limitations, hence a reduction in faults. Several researchers (Vanmali, Last, & Kandel, 2002), (Shahamiri, Kadir, & Mohd-Hashim, 2009) and (Yousif, 2015) have described that ANN is designed to replicate the structure and information processing of the brain. Instead of neurons, which are components of the brain, ANN's components are computational units. A neural network is from one or more layers that convey the process during input/output testing applications. Throughout this process, ANN can discover hidden knowledge and learn it during processing. ANN's can locate and learn during the input/output process of a testing application. ANN's have two formats known as *single-network oracles* and *multi-networks oracles*.

2.6.1 Single-Network Oracles

In general, ANN-based oracles began with a single-network oracle. Its functionality was retrieving the input data from the test applications to generate the ANN and result in the output data. Figure 2.3 illustrate a single-network ANN based oracle.

Figure 2.3 Single Network Oracle (Shahamiri et al., 2012)

As shown in Figure 2.3, the single-network oracle is with only one ANN or in other words a single neuron. A definition for each of the generated outputs is known as a single network (Shahamiri et al., 2010). In Figure 2.3, there are five outputs that the SUT has produced, due to the single-network oracle the outputs are generated all at once. This is a drawback and one of the major reasons why a single-network oracle is not efficient and not recommended. A single-network oracle is not efficient simply because, when it comes to a complex testing application, the ANN needs to learn the entire functionalities of the test case. Hence with the increase of the functionalities and complexity, the more likely that the ANN may fail. Therefore, emerges a different form of network oracle to solve this drawback and it's known as Multi-networks oracles.

2.7 Multi-Networks Oracles

Multi-network oracles are improvements over the single-network oracle, including eliminating the drawbacks contained in the single-networks oracles. The functionality of the multi-network oracles is similar to the single-network oracle except for the production of the SUT. In a single-network oracle, no matter what the number of outputs the SUT produced, the ANN generates the outputs together and causes a reduction in fault identification. More so, the ANN would not be able to generate results for a complex testing application. (Shahamiri et al., 2012) understood this problem and proposed a solution using Multi-networks oracles.

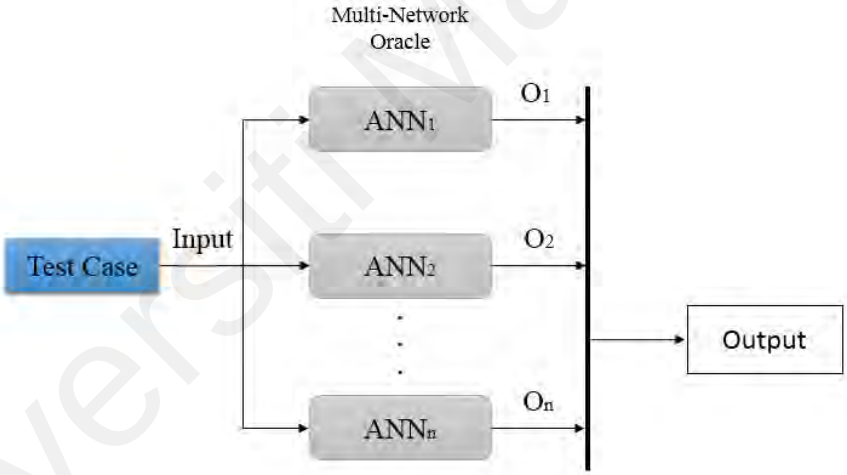

Figure 2.4 Multi-Network Oracle (Shahamiri et al., 2012)

As shown in Figure 2.4 the difference of functionality between the two network oracles is clear. As proposed by Shahamiri et al. (2012), the SUT in a multi network-oracle implies multiple single-network approaches in parallel. Instead of the outputs being generated all together, each output is separated and generated individually. With this method, the testing oracle can identify any faults in the application more accurately and more so able to generate complex applications. In addition, with the complexity of an application, and with

a multi-network oracle the SUT is cascaded among multiple ANNs components, each ANN has less to learn with less time and therefore simplifies the training process. However, multi-network oracles generally do contain some drawbacks though not so serious as compared to a single-network oracle and it is not highly taken into consideration by testers, this is due to the fact that most applications that are being tested are complex and may need more time to generate the faults of the application. Another issue is memory management, with the application being complex, it means that it is also large and this will require more memory to conduct the testing procedure to store the generated data. Nevertheless, multi-network oracle based on ANN has been a great contribution to tester.

2.8 Comparative study between Single and Multi-Network Oracle

An experimentation conducted by Shahamiri and colleagues (2012) constructed a case study and applied both test oracles using the same input and output vectors. They concluded with an evaluation result that the experimentation was able to find injected faults from over 3,000 executed and verified cases. Table 2.2 shows the evaluation results for both test oracles by Shahamiri et al., (2012).

Table 2.2 Single Network Oracle Evaluation (Shahamiri et al., 2011)

	Lower Threshold (Highest Precision)	Mid Threshold	Higher Threshold
Thresholds	Output 1: 0.08 Output 2: 0.015 Output 3: 0.04 Output 4: 0.02	Output 1:0.08 Output 2: 0.015 Output 3: 0.08 Output 4: 0.04	Output 1:0.08 Output 2: 0.015 Output 3: 0.08 Output 4: 0.08
Total Comparison		12000	
Average Threshold (Precision)	0.155	0.215	0.255

Total Absolute Error		**0.0206**	
Number of Injected Faults	9121	9064	8955
True Positive	2765	2866	3012
True Negative (Detected Faults) (Recall)	6275 (68.7% of the Injected Faults)	7549 (83.2% of the Injected Faults)	8008 (89.4% of the Injected Faults)
False Positive (Missed Faults)	2846 (31.2% of the injected faults)	1515 (16.7% of the injected faults)	947 (10.6% of the injected faults)
False Negative	114 (0.95% of the Total Comparison)	70 (0.58% of the Total Comparison)	33 (0.28% of the Total Comparisons)
Misclassification Error Rate	24.67%	13.21%	8.17%
Accuracy	**75.33%**	**86.79%**	**91.83%**

Table 2.3 A Multi-Network Oracle Evaluation results (Shahamiri et al., 2012)

	Lower Threshold (Highest Precision)	Mid Threshold	Higher Threshold
Thresholds	Output 1: 0.08 Output 2: 0.015 Output 3: 0.04 Output 4: 0.02	Output 1:0.08 Output 2: 0.015 Output 3: 0.08 Output 4: 0.04	Output 1:0.08 Output 2: 0.015 Output 3: 0.08 Output 4: 0.08
Total Comparison		12000	
Average Threshold (Precision)	0.155	0.215	0.255
ANN_1 Absolute Error		0.006	
ANN_2 Absolute Error		0.005	
ANN_3 Absolute Error		0.03	
ANN_4 Absolute Error		0.01	
Total Absolute Error		**0.014**	
Number of Injected Faults	9123	9139	9046
True Positive	2841	2851	2950
True Negative (Detected Faults) (Recall)	8094 (88.7% of the Injected Faults)	8874 (97.1% of the Injected Faults)	8841 (97.7% of the Injected Faults)
False Positive (Missed Faults)	1029 (11.3% of the injected faults)	265 (2.9% of the injected faults)	205 (2.3% of the injected faults)
False Negative	36 (0.3% of the Total Comparison)	10 (0.08% of the Total Comparison)	4 (0.03% of the Total Comparisons)
Misclassification Error Rate	24.67%	13.21%	1.74%
Accuracy	**91.13%**	**97.71%**	**98.26%**

Table 2.2 and 2.3 above shows the evaluation result from one of the case studies performed by Shahamiri et al., (2012). The experiments were conducted in three different precision levels withstanding from lowest to highest and applied the same vectors provided by the case studies. After the testing process, they concluded that the accuracy rate of the single-network oracle ranged from a minimum of 75% to a maximum of 91% whereas the multi-

network oracle a minimum of 91% to a maximum of 98%. Other than the accuracy rate, the experimentation also concluded that from the lowest to the highest threshold, the misclassification error rate dropped significantly for both test oracles but favored towards the multi-network oracle. They have deduced an improvement in neural network testing and mentioning that a comparison between those tests oracles and any other prominent oracles is required to sustain the challenges and in terms of quality will be beneficial.

Artificial Neural Networks is much more than just these two network oracles. The two network oracles shown above are just a process that used in ANN for a form of testing software behavior. It is much more complicated than just the diagrams above, ANN is a type of pattern recognizers that has to learn the application and process and more so the network has to be trained in order to apply the testing process. The training process is held before the oracle is tested with any proposed solution. Since it clearly shown that multi-network oracle is much more efficient, this takes more time for the training process due to the fact that each SUT is separated to each ANN, which means that a learning and training process is being conducted on each ANN. Furthermore, these processes are conducted over and over again until adequate error is achieved. This simply means that the entire testing application process is complete and if it were to fail that would mean that the application is simply too large and complex which generally occurs in a single-network oracle.

2.9 Info-Fuzzy Networks

Info-Fuzzy Networks (IFN) carries a similar form of application to ANN. IFN is adequate on knowledge discovery and data mining methodology and is concerned in the combination of feature selection and classification. The difference between ANN and IFN

is that the latter does not carry decision making process. Firstly, an explanation on what IFN contains and that is feature selection.

In a software testing approach, feature selection indicates the process of identifying the components of the input component which is a crucial segment in the decision process (Agarwal et al., 2012). With IFN, there is an interaction between the inputs and outputs that are represented as the multilayered decision graph generated by the IFN (please refer to section 1.2).

There have been several attempts of using IFN to make test oracles automatically. One of such was conducted by (Vanmali et al., 2002). They introduced the functionality of IFN on an automated black-box regression tester, where a representation of theoretical information is coupled between the input and the target attributes. The automated approach proposed by these authors was functional and was a multi-layered network oracle similar to the approaching Shahamiri et al., (2012). Using IFN approach, each input attribute was divided to generate individually, associated with a single layer and produced an output, extremely similar to the multi-network oracles applied with ANN (Shahamiri et al., 2012). An automated approach was developed that was able to generate, execute and evaluate the generated output data automatically based on regression testing (Vanmali et al., 2002). Even though the automated approach developed by them was a success, they did not explain the drawbacks and faults of their research.

Shahamiri et al. (2012), showed the drawbacks of IFN as an automated test oracle. The differences between the two applied automated oracles were simple. ANN was superior due to the capability of being able to handle complex application while the proposed IFN automated black-box regression testing was unable to study and learn complex application,

more so solve a partial differential equation. In addition, the multi-network formed by the IFN is less accurate than a single-network approach even though a multi-network approach is supposed to function more accurately since the operated input applied and generated individually allowing faults and errors to be detected, whereas a single-network processed all the input together at once resulting in ~~a~~ difficulty in detecting the maximum faults and hence being a less accurate oracle. In conclusion, IFN based oracle can function and generate the necessary output despite its drawbacks and applying a multi-network ANN oracle is more applicable in obtaining the necessary results.

2.10 Support Vector Machine

Support Vector Machine (SVM) is one of the latest pattern recognizers in the area of pattern recognition. SVM has been known to be the pattern recognizer that can improves the classification performance and becoming one of the developing tools for machine learning and data mining with the capability of performing both classification and regression. SVM is a learning system applied mainly in algorithmic formats that uses a linear functionality in high dimensional feature space. In other words, SVM has been introduced with the capability of solving functional estimated problems and pattern recognition. SVM is based on the Vapnik-Chervonenkis dimension which is structured to show that the generalization error is bounded by the total of the training set that is being used (Cortes & Vapnik, 1995). This dimension is a computational or statistical learning theory that's measures the capacity of a classification algorithm.

The pattern recognizer SVM is a tool that came after popular recognizers such as ANN and IFN and most researchers that have conducted experimentation on SVM and have had a

similar general conclusion that SVM is an easier tool to understand and use. Unlike other machine language methods, SVM generalization error does not correlate with defected parts of the input data and this is vital to SVM provided a good performance when processing in high dimensional complex applications. SVM represents the input data into a high dimensional characteristic space in which an optimal separating hyper-plane is constructed. An optimal separating hyper plane is a linear classifier with the maximum margin for a given set of learning patterns. Figure 2.5 illustrate how an optimal separating hyper-plane is.

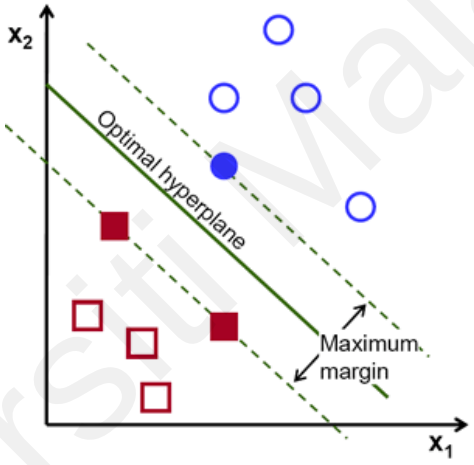

Figure 2.5 Optimal Separating Hyper-plane (Cortes & Vapnik, 1995)

In SVM, the optimal separating hyper plane separates the input data and constructs an optimal hyper plane which is an empty space between the inputs. This space allows the pattern recognizers to not relate to any problems that are conjured with the input data. On the other hand, it also allows for a high capability of generalization utility. This generalization method will be able to process each data for error detection before generating output automatically. This is how SVM is capable of overcoming an over fitting

problem experience, problems that other machine learning methods such as ANN are not able to solve.

SVM does unfortunately contain a minute drawback that causes the approach to generate minor inaccurate data, it also affects the performance of the pattern recognizer, the problem is known as a non-linear problem. However, researchers throughout the years have developed a solution to this problem and in addition increased the accuracy and performance of an SVM application, known as Fuzzy Theory Support Vector Machine (FSVM).

2.11 K-Nearest Neighbor

K-Nearest Neighbor (KNN) classification is one of the simplest classification methods and is usually the first choices for classification study when there is little or no prior knowledge about the distribution of the data(Peterson, 2009). KNN rule identifies the category of unknown data sectors on the basis of its nearest neighbor whose class is already known (Bhatia, 2010)

Even though KNN is less than a high standard or quality pattern recognizer, research has shown that in the infinite sample solutions, KNN provides a drop of the error rate when used against small problem domain characterized by data in classification (Keller & Gray, 1985). Although KNN contains an advantage in classification due to its rather computational simplicity and generating good results, the pattern recognizers contains drawbacks such as not being able to compute large and complex software's and overlapping between the input vectors and the sample data by considering both vectors as equal.

2.12 Literature Review Findings

Based on the literature review findings, firstly we discussed about software testing activities. These methods determine testers the challenges that need to be addressed before moving to the next phase. Secondly is software paradigm's, this determines the form of testing that is required to be executed. A Software paradigm acts as a platform for software testing. Commonly used is the black-box approach since this paradigm's testing methods are solely focused on the function of the modules. The main purpose is to verify the input/outputs of the program compiling with the specifications and not concerning with the actual structure of the program (Agarwal et al., 2012). Lastly is pattern recognizer's which is types of tools that are implemented in software testing processes. The use of pattern recognizers is to reduce software faults in a test by using tools such as ANN, IFN and many more, that are machine learning tools focusing on data mining, misclassification, clustering etc.

2.12.1 Challenges in existing test oracle

Based on the literature review findings, two notable pattern recognizers applied as software test oracles are mentioned. Below are the existing challenges of these test oracles.

1. Single-Network Artificial Neural Networks (ANN)
    - Is one of the most successful formats of software testing oracles due to its benefits being greater than its limitations (Shahamiri et al., 2010).
    - In an ANN as a single network oracle, this pattern recognizer it requires an expected outputs to be generated manually (Shahamiri et al., 2010).

- Unable to process complex applications and generate precise calculations (Shahamiri et al., 2010).
- Requires additional information and training for the ANN modeling (Shahamiri et al., 2010).

2. Multi-Network Oracles based on ANN
    - As eliminated all of the challenges in the single network oracle (Shahamiri et al., 2012).
    - However due to the framework of the multi network where each input will be generated in a single ANN (Refer to figure 2.4), this causes the process to take up more time, to executed the test data and generate outputs (Shahamiri et al., 2012).
3. Info-Fuzzy Networks
    - This test oracle requires to be restructured before the next test; hence it is unable to be used as a fresh test. This cause time consumption to the testing process (Shahamiri et al., 2012).
    - Info-Fuzzy Networks is limited to only regression form of testing (Shahamiri et al., 2012).
    - It is unable to test new functionalities (Shahamiri et al., 2012).
    - Requires a reliable legacy system (Shahamiri et al., 2012).
    - Requires additional knowledge to the IFN modeling (Shahamiri et al., 2012).

There have been a dozen researches on SVM, mainly on the classification methods, fuzzy theory classifiers and based hybrid classifier, but so far there has not been any publicized paper on SVM applied as an automated test oracle. This research may be the first and the

objective of this research is to know if SVM is capable of being an effective tool to be applied as an automated test oracle and more so whether it functions better than previous pattern recognizers.

Pattern recognizer is a form of tool for generating processes such as classification, clustering, neural networks etc. when conducting a testing process. Whether it is software behavior, data mining or classification of data, the end point of an automated test oracle is to take the input to generate the end results output and to seek the best fault report for the application. This research begins with the understanding of the functionality and capabilities of SVM and the comparison with other pattern recognizers such as ANN and IFN. Similar to ANN, SVM is a tool that trains to understand the application before beginning the testing process and generating results. The objective of SVM tool described by is to compare two-group classification problems, implementing the input vectors which are non-linearly mapped to a very high dimension feature space. In this feature space a linear decision surface is constructed. The SVM's ability in implementing a restricted case, where the training data can be separated without errors. Due to the fact that SVM is trained with a leaning algorithm and capable to construct an optimal separating hyper-plane, it is deduced that SVM can handle very complex application. Although there are drawbacks of SVM, they were solved by applying a fuzzy theory tool known as FSVM, the factor that SVM tool is capable of complex application and can generate data in a small medium such as the optimal hyper plane is an advantage.

Table 2-4 Automated Test Oracle Comparative Study (Shahamiri et al., 2011)

Oracle Model	Test approach	Addressed challenges	Remarks and Limitations
Human oracles	None	Benefit: - Constraint: - • Output domain generation • Mapping • Comparison	1. Un-automated 2. Very expensive 3. Time consuming 4. A human oracle must completely understand the software domain
Random testing	Black-box	Benefit: - Constraint: - • Output domain generation • Mapping • Comparison	1. Un-automated 2. Unreliable 3. Low cost 4. Imperfect oracle
Cause-effect graphs	Black-box	Benefit - • Mapping Constraint - • Output domain generation • Comparison	1. Automation requires specification or code analysis tool to fetch the logical rules. 2. Expensive in case humans considered to provide the logical relationships and required graphs 3. Can lose readability in complex applications
IFN regression tester	Black-box	Benefit - • Output domain generation • Mapping • Comparison Constraint -	1. Cannot be used for a fresh test. 2. Only applicable in regression testing. 3. Cannot test new functionalities. 4. Requires a reliable legacy system. 5. Requires addition knowledge for IFN modeling.
Single-network ANN oracle	Black-box	Benefit - • Mapping • Comparison Constraint - • Output domain generation	1. Requires manual expected output generation 2. It could not be reliable to test complex software and precise calculations 3. Requires additional knowledge for ANN modeling 4. Less expensive in case the output domain has already been provided
Multi-network ANN oracle	Black-box	Benefit - • Output domain generation • Mapping • Comparison	1. Fully automated 2. Can be used for a fresh testing 3. Can evaluate both new and old functionalities 4. More reliable than traditional Single-network oracles 5. Less expensive

Chapter 3

3 Research Method

In this chapter, the methodology of study us explained, followed by the proposed approach, design and procedure that is being applied. It also describes the design and finally the experimentation process and evaluation criteria that is considered in this research

3.1 Methodology of Study

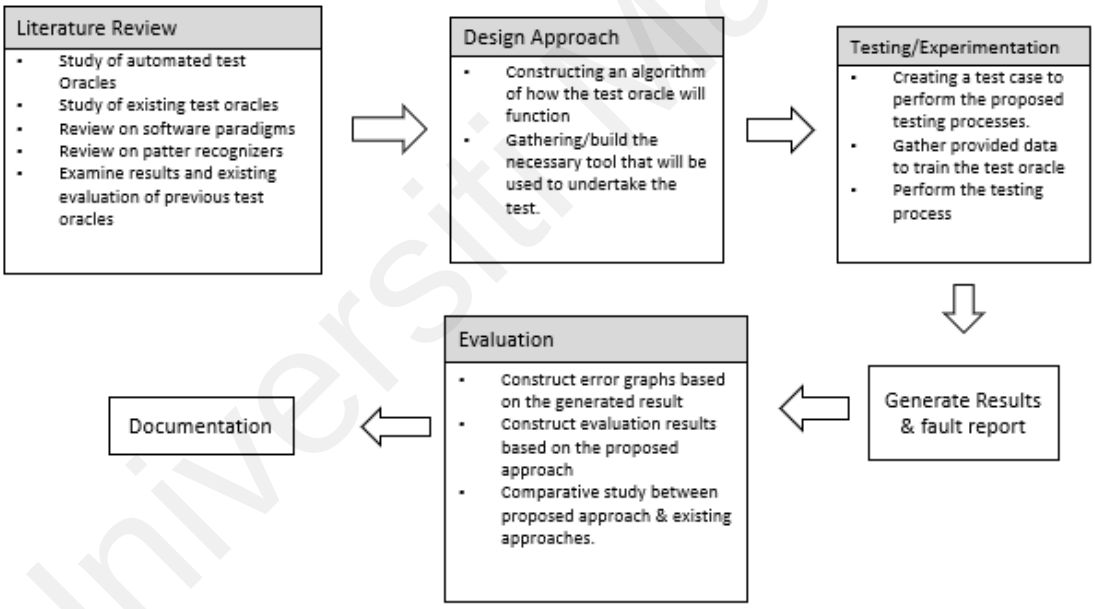

Figure 3-1 Methodology of Study

A few methods of study are used in order for the research to begin. Since this research is about automated software test oracles, which is a practical research, discussing on how software testing oracle can be improved and become more reliable for testers. Various

researches have been conducted on automated test oracles but more specifically, research has been conducted on the method that will be used for this current dissertation, including pattern recognizers such as *ANN* and *IFN*. Beginning with literature review, researching existing articles/journals and conferences contemplating about the related topic of the dissertation. Furthermore, this method is usually conducted by library research and sites such as scholar.google.com, ISI journal website etc. On specific terms, beginning by analyzing previous journals about automated software test oracles, I/O relationships analysis, Support Vector Machines, approaches and methods that is the most relatable and instance to the current topic.

During the design phase, the existing approaches to test oracles are analyzed to determine the suitability and the limitations of the existing pattern recognizers. The design phase enables the author to select the appropriate pattern recognizers for the testing/experimentation step.

In testing/experimentation phase, the selected pattern recognizers will be tested, and the results generated by each pattern recognizer enable the author to conclude whether the pattern recognizer is a success or not.

The outcome from the testing/experimentation phase, the data is compiled, analyzed, evaluated, and a conclusion is drawn. Finally, all the research method phases are documented.

3.2 The proposed approach

3.2.1 Support Vector Machine (SVM) as an automated test oracle

The proposed SVM approach, which is based on the Single Network Oracle (Shahamiri et al., 2012) will be explained in this section, including how this test oracle will function and generate the provided data. On top of that, the proposed approach will be compared for its reliability and usage in comparison with other test oracles.

As mentioned in, section 2.5 ANN can generate a more accurate and qualified data result when applied in a multi-network oracle. The multi-network oracle separates the input data into its individual ANN and this improves the performance of the testing application and detects more faults in the applications to generate a qualified report. Like ANN, SVM constructs a similar format to operate on complex high volumes application and able to retrieve results. This of course compared to a multi-network oracle is much more less time consuming and requires lower memory management. Figure 3.2, shows the Single Network Oracles framework (Shahamiri et al., 2010) of an automated test oracle base on SVM.

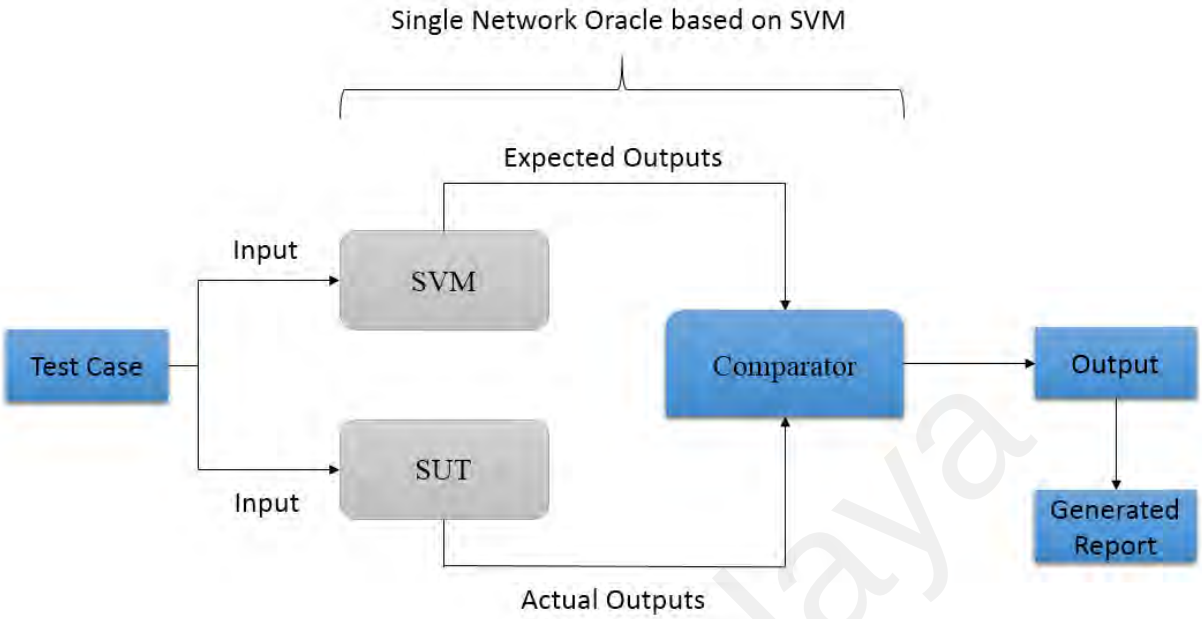

Figure 3.2 A SVM approach based on Single Network Oracle Framework

Figure 3.2, shows the SVM framework of the test oracle on how it functions during software testing. After the SVM is trained and has learned the data, it will then able to perform the testing. The process is practical under the software paradigm black-box approach. The data is inserted by the test case, which is then processed through the SVM and SUT. The SUT functions is to generate the actual outputs. The SVM maps the input data from the test case into a high dimensional characteristic space and an optimal separating hyper-plane is constructed. The optimum hyper-plane is able to separate the problem of the input data into two dimensional spaces. The space in between the problems is what's called an optimal hyper-plane, this allows no generalization error to relate to the problem, and allows high generalization ability to function. This is the function that allows SVM to generate complex applications. Nevertheless, existing body of knowledge has grasped that a multi-network oracle is more efficient, accurate and better in performance.

Essentially, SVM is one of the recognizer where, its functionality may achieve automated testing oracles as a better automated tool. SVM, unlike ANN, does not interact with the related problem in an input data(Cortes & Vapnik, 1995). In addition, SVM constructs an optimal separating hyper-plane which does not consist with another pattern recognizer (Cortes & Vapnik, 1995). Optimal separating hyper-plane is the reason why when applying SVM as an automated test oracle, it is able to function in a single-network oracle (Cortes & Vapnik, 1995). Table 2.4 shows the comparison between previous test oracles all under the same test approach, after the experimentation of the proposed approach, the evaluation will be conducted. Only two will be explained alongside SVM in this research which is ANN and IFN. This is simply due to the fact that these three uphold a similarity in mapping, comparison, output domain generation and classification, all represented in a black-box paradigm. Another question is concerning automated test oracle based in SVM is the performance. Since all the input is generated at once, this must cause a flaw in the accuracy and performance of the test oracle. This is not the case, the factor is that applying these test oracles concentrates solely in generating the data and not the problems that the data provides, and in addition, the fuzzy theory tool to will be added to the pattern recognizer that will discontinue any form of noise disturbance towards the data. This allows of the testing process to maintain its accuracy and performance and finally, since the framework of the test oracle only implicates a single-network oracle format, the test oracle is not stressed when performing generation of data hence the performance is not manipulated.

3.2.2 Experimental Design

The objective of the experimentation process is to use the test oracle with the data provided and generate the end result. In this research, three different experimentations will be

conducted and the performance of each experiment will onboard a comparative study. Each experiment will be run twice with a clean and faulty data (Shahamiri et al., 2012). The clean data is to justify the optimal results of the test oracles. The faulty data is to identify how the test oracles will handle such data and whether the process will continue to generate a fault report (Shahamiri et al., 2012).

3.2.2.1 The Classifiers as test oracle approaches

a) Support Vector Machine

The first experimentation will be executed as an SVM single run. This run is based on the process shown in figure 3.2. Since SVM can deal with complex applications using its constructed optimal separating hyper-plane, it will ignore any faulty information and focus on the correct data. This feature allows the classifiers to work as a single network oracle.

In this research, WEKA toolkit version 3.6 is used for conducting the first experiment. WEKA toolkit is a tool developed to access variety of machine learning approaches including SVM for the purposes of experimentation and comparison using the real-world data sets. WEKA is selected because it contains flexibility on machine learning algorithm, and is much simple to understand and execute.

b) SVM and KNN

The second experiment in this research make use of experimentation is using SVM with KNN. The reason KNN was used is because it is compatible and similar to SVM in forms of classification and regression. In addition, KNN fits

perfectly with the proposed courier industry as KNN is commonly used when it comes to distances and data obtained from maps

c) ANN

The third experimentation is ANN, one the most widely studied and used pattern recognizer in software testing when implicated to software behavior. ANN functions as an automated test oracle and show the results based on past research is discussed in Chapter 2. Therefore, to obtain an accurate comparative result, the exact same test and training data for the support vector machine were used for ANN.

3.2.2.2 The Experimentation Process

The flow of the experimentation process is as follows.

a) Selected domain

This research domain is the Courier Industry. The domain would be used on both classifiers, in order to obtain a comparative evaluation. The courier company in focus is DHL.

b) Data collection and Preparation

The data is provided based on the case study which is a courier company that focuses on time prediction window application, where the courier company is able to predict the time of shipment delivery. On a note, there is no ceiling for an amount of dataset required for a software test oracle to generate. Generally large amounts of data are able to withstand processing, the differences would be on the time taken to process and generate a result which will be showed in chapter 4. There are two sets of data

that needs to be provided for the testing process, which are the master data and the input data. The input data will be fed into SVM as an automated oracle, while the master data is the base source of information of comparison with the result before generating the final output. In addition to the input data, in order to understand what may be the limitations of SVM and the classifiers, the experimentation will be run twice with clean and faulty data.

a) Training the data: Before the experimentation process begins, the test oracle will be trained towards each test oracle which is SVM, SVM+KNN and lastly ANN. The reason for using only those three test oracles is because these test oracles are:
    a. Reliable for complex testing
    b. Able to be reused as a fresh test.
    c. Address challenges automatically.
    d. Cross platforms compatibility and not platform dependent.
    e. Does require a legacy system in order to train the data.
    f. Able to evaluate old and new functionalities.
    g. Fully automated.

    The data provided will be used to train the test oracle, mainly for the pattern recognizers to obtain the best fault detect result (Shahamiri et al., 2012). The advantage of using this proposed approach is that the test can be restarted; therefore, several runs can be conducted. Two classifiers would be tested. One is the SVM classifier and another is K-Nearest Neighbor classifiers.-The aim of the training to allow the SVM to notify with the input and output data in other words an

input/output relationship analysis. A tool that will be used for training the SVM known as *LibSVM using JAVA.*

b) Testing Process: After the test oracles has been trained, the testing process begins by following the approach shown in *Section 3.2*, *Figure 3.2* and the test oracle will proceed with the process and generate a finalized report.

3.3 Evaluation

The evaluation of the proposed approach is conducted based on the generated result from the experiments. This evaluation analysis categorized based on the following

1. Performance and time consumption – these two attributes are tied together as measured based on the run time of the experimentation. The run time dictates the quicker the run process to generate an end results concludes a greater performance of the experimentation as well less time consumption. Since testing run time varies due to different methods of testing, there is no fixed benchmark for a positive performance rate and run time. However, for the following experimentation, the benchmark will be upholding accordingly to the ANN experimentation by Shahamiri et al. (2012) as shown in Table 2.2 and 2.3.
2. Accuracy and Error rate – these two attributes are tied together as measured based on the comparison result between the master data and input data. The input data is cross referenced against the master data, with the generated result as follows

   Number of matched input data/ Total records of input data

   The statement above indicates the generated result will contain how many records of the input data is accurately cross referenced against the master data. This results also

concludes how accurate the testing process is by the number of matched input data, this is based on the exact matches. Likewise, for the error rate as the system calculates the false matches. The lower false matches equal the deterioration of error in the testing process.

From the experiments, tables are generated to show the consistency of the classifiers in fault detection. Most importantly for the evaluation is the proposed approach evaluation results which consist of the number of testing undertaken by the proposed approach, the error rate and the accuracy. Comparative study is conducted between the generated results of each classifier, along with all the necessary categorical analysis and existing test oracles that have been tested on well accepted data (Shahamiri et al., 2012). The reason why it is essential to conduct a comparison against ANN, is simply because ANN is a widely-studied pattern recognizer (Shahamiri et al., 2012) and contains similar functionalities with SVM such as classifications, regression and input/output data analysis. This comparative study will be able to show the result differentiation within the proposed approach and other existing pattern recognizers using the same case study and generated data. The performance evaluation is similar to existing evaluation processes proposed in (Shahamiri et al., 2012).

Chapter 4

4 Experimentation

In this chapter, experimentations are performed to validate the classifiers performance (benchmark and proposed approaches) and are explained step by step, beginning with the selection of courier domain and the different types of data classes used and the experimentation. In order to improve the accuracy of each classifier, different classifiers such as SVM and K-Nearest Neighbor are applied for this test. The purpose of this experimentation is to analyze and obtain an understanding of how accurate is SVM as an automated test oracle can be, the performance results and more importantly how the testing method is able to handle fault data.

4.1 Domain and Data Selection

For the experimentation and evaluation purpose, the data is from DHL, a courier company. The experiment focuses on time prediction window application, where the courier company is able to predict the time of shipment delivery. Prior to shipment being delivered as the delivery time is predicted, the company can notify their customers in advanced via text message or email, informing the customer exactly when their shipment will be arriving (e.g. between 10:00AM-12:00PM).

The data for the experiments contain criteria's such as Geocodes, postal codes, ready time, closing time and most importantly, time of delivery. In a world of a courier's task, each courier is provided a dedicated route (a route contains multiple postal codes). The courier staff will drive on this route on a daily basis delivering shipments. Courier companies have applications to obtain data of when the courier staff delivers the shipments,

but most importantly at the end of the day, it will result in a pattern of how the courier staff drives from one stop to the next. The following day the courier applies the exact same task, on the same route and makes the same stops. From this, it is possible to take this pattern and execute a time prediction window.

The data has an I/O domain, where the selected criteria of data is run as input, the pattern recognizer will filter all the faulty data and provide output that accurately state when a shipment will be delivered in a certain Geocode at this particular time every day.

4.2 Data Preparation

Preparing data is a crucial task for experimentation, where two sets of data are used, the master data and the input data. The master data contains 45,757 records of data containing geocodes and time windows. This master data will be used as the comparator for the input data.

The input data contains data of single courier's daily delivered shipments. In a real scenario, the data is immensely faulty due to the fact that data is provided by humans and hence contains human errors. Therefore, obtaining a completely clean data between the master and input data is difficult. The key of the experimentation is to obtain different results for a comparative evaluation, as such clean data and faulty data are both required. Before training the master and input data, information that may contain high probability of human error such addresses, customer name and contact numbers are removed for the data. Duplicated data are also filtered out, as duplication are unnecessary and have no impact to the result other than being a load that will increase the run time. As a result, after cleansing and filtering the faulty data, 45,757 records were obtained. Table 4.1 shows the sample of

master and input data taken from DHL. The geocodes columns are the key identifier to be compared with the master data. Geocodes are the most accurate data, containing the longitude and latitude of when the courier made a stop and delivered the shipment.

Table 4.1 Example of Master and Input Data

Geocodes		**Actual Delivery Time**	**Time Windows**
Longitude	Latitude		
52.29465	4.768664	14:05	13:00-15:00

4.3 The Experiment

The proposed oracle was applied to verify the data from DHL. The application builds to run these test data's, using java programming on IntelliJ IDEA application, functions as the following:

1. The application will read the input data from the input table.
2. The input data will flow and processed by the functionalities of the provided classifier and generates an output.
3. The output would then be compared with the master table. In this comparator, three factors will be calculated, which are the exact match, correct match with potential windows and the wrong windows.
4. The output is then being generated as a report.

4.3.1 Classification using WEKA Toolkit

Developed in the University of Waikato WEKA is a toolkit developed to allow users to access a variety of machine learning approaches such as SVM for the purposes of running

testing processes and conducting comparative studies using real-world data sets. WEKA currently runs with machine learning tools written in a variety of programming languages such as C, C++, LISP and JAVA. WEKA is not a single program but rather a set of tools bound together by a common user interface. (Geoffrey Holmes) (Garner) (Robert J. McQueen)

Below are the commands using WEKA toolkit based on JAVA

```
public class TWPredictor {
        public static void main(String args []){
                try{
                        BufferedWriter writer = new BufferedWriter(new
                        FileWriter("C:/Users/Muhammad/DesktopjavaML
                        lib/ReportAlgo_res.txt"));
                        /* Load a data set */
                        int classIndex = 3;
                        Dataset data = FileHandler.loadDataset(new
File("C:/Users/Muhammad/Desktopjava ML lib/twRawDataBoth.csv"), classIndex, ",");
*/
                        // decision.
                        SMO smo = new SMO();
                        Classifier weka = new WekaClassifier(smo);
                        weka.buildClassifier(data);
```

4.3.2 SVM Based on Single Network Oracle

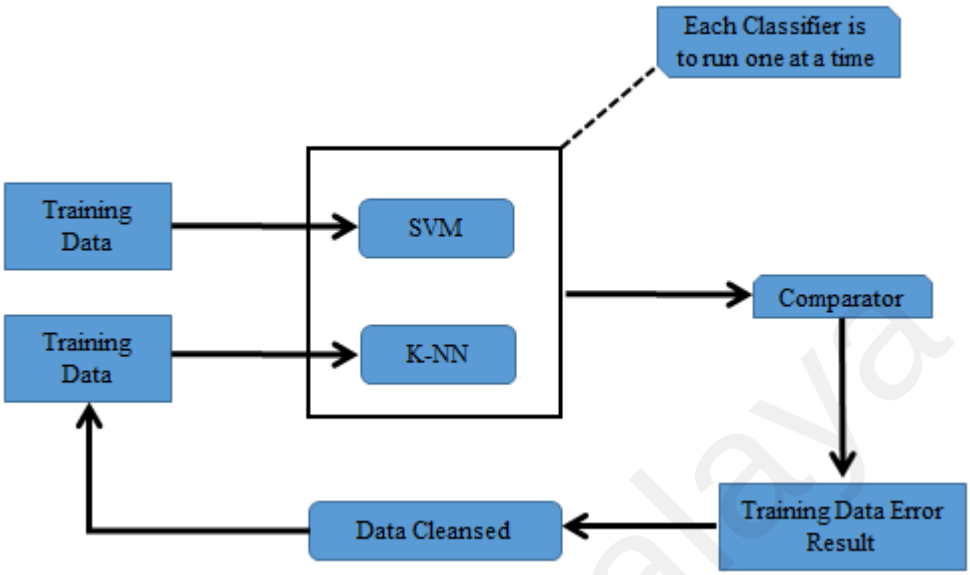

Figure 4.1 Training SVM Based on Single Network Oracle Framework

Before the experiment can be conducted as a machine learning process, the classifiers need to be trained. The purpose of training is to ensure that adequate error rate is reached. Note that the value of the error rate depends on the amount of training data that is being used. As show in figure 4.1, the training data will be inserted as input data into the network oracle SVM twice as two different SVM classifiers being used. Each classifier is provided a set of training data. Once a cycle is complete, a training data result is generated. If the error rate is too high, the process of preparing the data is repeated as more cleansing is required. This training process also guides pattern recognizers to practice, understand and familiarize with the process hence the process is implemented multiple times. Once the training process is complete, the classifiers are trained; and can be applied for testing process. The case study is executed on the trained classifiers and the system under test (SUT) simultaneously. The test cases are given to the trained classifiers which also being executed on SUT. Actual outputs

are generated by SUT, which expected output is generated by the SVM oracle which contains each classifier. Both outputs are then compared with the master data which then generates a final output result. Note that the entire process contains minimal human effort for the preparation of the environment, nevertheless the rest is automated.

4.3.3 Dual Classifier

KNN is an algorithmic classifier used in pattern recognition. It is a non-parametric method used in regression and classification, and a very simple approach that is effective and efficient among pattern recognition.

KNN, just like many other pattern recognizers also comes with limitation as follows (Suguna & Thanushkodi, 2010)

1. High calculation complexity: To identify out the k-nearest neighbor samples, all the training samples must be calculated. When the training sample is not efficient, the KNN will not be optimal. A solution to this problem would be to reduce the dimension of the feature space using smaller datasets and improving the algorithm.
2. Dependency on the training set: The classifier processes the training data and it will not apply additional data. This result in a dependency every time there is a small change applied on the training data, the classifier needs to run an entire recalculation.
3. No weight difference between samples: All the training samples are generated equally; there is no differentiation between minute and large data. This is to

avoid a mismatch in the actual phenomenon where the data samples have an uneven distribution commonly.

In the end, K-NN is still known to be the most important non-parameter algorithm and it is a supervised learning algorithm.

Below are the commands using SVM+KNN.

```
public class TWPredictor {

    public static void main(String args []){

        try{

            BufferedWriter writer = new BufferedWriter(new
            FileWriter("C:/Users/Muhammad/Desktopjava ML
            lib/ReportAlgo_res.txt"));
            /* Load a data set */
            int classIndex = 3;
            Dataset data = FileHandler.loadDataset(new
            File("C:/Users/Muhammad/DesktopjavaML
            lib/twRawDataBoth.csv"), classIndex, ",");
            Classifier knn = new KNearestNeighbors(25);
            knn.buildClassifier(data);
```

Chapter 5

5 Experimentation Results

This chapter explains the experimentation results of the classifiers, which are SVM, dual classifier SVM with KNN, and ANN on clean and faulty data. The aim of the experiment was to run three different types of classifiers based on the proposed solution as shown in chapter 3, using the same input data and master data. There are two different sets of data used in this experiment, the cleansed data and the faulty data.

With all the experimentations conducted and results generated, this chapter will discuss what the results mean and its implication. A total of six experimentations (two for each classifier) were conducted with numerous runs of training processed. The first experimentation was aimed to show the experimentation run using cleansed data, to show the accuracy rate and run time of the test. This was conducted by building an application, training the oracle and running the experiment to generate a result. The second experimentation was aimed to show how the oracle will handle faulty data. This was conducted by using inaccurate, duplicated faulty data. The oracle was trained and running the experiment to generate a result. Both experimentations follow the framework that was shown in *chapter 3 section 3.2* and trained the oracle following the framework in *chapter 4 section 4.3*. Lastly this chapter will also show a comparison between the proposed method and an existing method ANN. This outcome will be able to show us a benchmark on where SVM stands to be used an automated test oracle.

5.1 Experiment 1: Using Cleansed Data

Both classifiers were trained and generated using cleansed data. Cleansed data is data that has been filtered to contain any faults; this is to ensure the generated test to obtain an accurate result and error free.

5.1.1 SVM Classifier

In this first experiment, it is conducted on SVM classifier using cleansed data. This clean data consisted of removing data that may contain high rate of human errors such as address, contact numbers, postal codes etc. The key identifier was using latitude and longitude data as that was the most accurate, showing the exact location of a courier delivering an item. After the data was cleansed, the next processed was to train the oracle, as mentioned earlier, the experimentation is based around pattern recognizer on machine learning, classification and training is required in order for the oracle to understand the process of an input/output data. The training was conducted by inserting the training data separately to each classifier. The classifiers have different functionalities but are put through the same testing process, with the same data and to generate the same format of results. The training data is inserted and generated based on the functionalities of the classifiers and generates an output the output is then compared with the master data that consist above 45,000 records. During the process, each line of input data is compared with the master data to find the exact matches, false matched and potential probabilities. Based on the accuracy and the error rate of results, the training data goes through cleansing and trained multiple times.

After the training is completed, we began with the main experimentation following the framework in the proposed method (Chapter 3, Section 3.2). Firstly, when using a single SVM classifier, we evaluate that this classifier has a faster in reading data, running the algorithm and generating the result. As it's processed, this classifier displays its optimal separating hyper plane by mapping the input data and separating that data to create the empty space in this linear classifier. This is where the exact matches and probable possibilities from the input data was put aside and the errors were ignored. This process provides an accurate result as the patter recognizer is not manipulated by errors and focuses on the correct data.

The process of comparing with the master data took 450 records as exact matches over 1,610 records. The result is not an acceptable result and this is due to one drawback, SVM classifier is unable to calculate probable possibilities. Probable possibilities are input data that is compared in the master data as exact matches but in multiple times. This simply means that one line of data from the input data has a possibility of more than one exact match. This classifier fails to read multiple results and concludes it as false matches. However, on a positive aspect, it is concluded that the testing process can generate correct data and ignore false data completely.

5.1.2 Dual Classifier (Clean Data)

Next part of these experiments is to use the dual classifier of SVM and K-Nearest Neighbor, which provided a more positive result. The same exact process was applied similar to the single SVM classifier. The difference falls in the use of K-NN together with SVM, which is more accurate as compared to SVM. Although the run time in reading the

data, processing the comparison and generating the result is much longer, SVM+K-NN classifiers able to generate 850 exact match records out of 1,610 records with 752 false matches and eight probable possibilities as a match. The classifiers were able to generate an optimal hyper plane, mapping the input data and creating an empty space, and ignoring the false data. When comparing with the master data, the classifier used for classification and regression, calculates K=1 averaging the nearest neighbor, in this case the probable possibilities and generates exact matches. The eight potential possibilities were due to too many possible times for a certain input data, this result to the classifier not able to classify the comparison and generate an exact match. However, the classification process takes into consideration that there is at least one time that it's an exact match, but due to multiple different times, the data is placed as a potential possibility. With clean data being used the SVM oracle functioned as expected, SVM + K-NN being a more positive tool to use in an automate test.

5.1.3 Summary of the First Experiment

In conclusion for the first experimentation, the dual classifier (SVM+KNN) performed much adequate as compared to the Single SVM. Although the dual classifier's run time took slightly longer, the classifier was able to generate twice more an accurate result with a reduction of false data. In addition, the classifier was able to generate potential probabilities, indicating the ability to read similar time windows in the input data when comparing with the master data and since the number of similarities is more, it will be placed as a potential probability.

Table 5.1Summary of the First Experiment

	Single SVM WEKA Classifier	Dual Classifier (SVM+KNN)	Percentage of differences
Master Data	45,757 records	45,757 records	0%
Input Data	1,610 records	1,610 records	0%
Run Time	8s 50ms	1m 12ms	-607.29%
Exact Match	420 records	850 records	102.38%
Potential Probability	0 records	8 records	NA
False Match	1,190 records	752 records	36.81%

5.2 Experiment 2: Using Faulty data

In this second experiment the exact same process as the first experimentation was applied, the only difference is that it's using faulty data. The aim of applying this experimentation is evaluate how the test oracle handles faulty data when training and running the process and if the oracle is able to maintain the process of not relating to the problems of the input data. Normally to execute this type of experimentation faulty data is required and this can be done by

1. Manipulating the data by injecting faulty errors.
2. Data that is contained to have high rate of error due to human error such as addresses were applied.

The expectation is that the injected errors will be completely ignored. If not, the application will simply prompt an error and will not be able to continue to run. Also applied is using the original data provided and not the cleansed data, since it was generating by humans, we decided not to tamper with the data and use it, this way a result can be generated based on data that is produced and manually written by humans and not automated.

Next the training process is running multiple times and finally the main test. The expectation was a higher false match will be obtained. After the run was completed, the first

conclusion was that no error was prompted in the application and the system continued to run until completion. This of course tells us that the injected errors were ignored and not taken into consideration.

On the other hand, using the original input data provided there was a large difference in the results compared to the cleansed data.

5.2.1 SVM Classifier (False Data)

The single classifier SVM is conducted using false data. The result generated showed a margin of difference compared to using clean data. The SVM classifier still maintained a quick run time in the case of reading, processing the comparison and generating the results. However only 277 records generated were an exact match out of 1,610 records. This resulted in a false match of 1,333 records. Once again, the inability of the SVM classifier alone to calculated potential possibilities resulted in a greater false match.

5.2.2 Dual Classifier (Faulty Data)

The dual classifier (SVM+K-NN) on the other hand, still maintained as a better classifier to the SVM alone. The run time was faster compared to using clean data, that is because the data contains more faults and hence the classifier simply ignores those records. This results in the process to generate much faster. The same exact data was used and the dual classifier resulted in only 392 exact match records, false matches of 918 records and 300 records of potential possibilities. As expected the exact matches are to decrease, however with the dual classifiers ability to match similarities in the input data and generate potential probabilities, the classifier did not discard 300 records, instead it acknowledges that 300 records contain a

nearby range of Geo Codes and similar time windows and hence these records have a potential possibility to be used.

5.2.3 Summary of the Second Experiment

Based on this experimentation, we can evaluate that the process still maintains its framework using the classifiers. The data that is considered faulty when processed in the oracle separated the correct input data and ignore the false data. Nevertheless, the dual classifier still maintained as a superior classifier against the SVM classifier.

Table 5.2 Summary of the Second Experiment

	Single SVM WEKA Classifier	Dual Classifier (SVM+KNN)	Percentage of differences
Master Data	45,757 records	45,757 records	0%
Input Data	1,610 records	1,610 records	0%
Run Time	7s 48ms	45s 46ms	-507.75%
Exact Match	277 records	392 records	41.52%
Potential Probability	0 records	300 records	NA
False Match	1,333 records	918 records	45.21%

5.2.4 Experiment 3: Comparative Study against ANN

As mentioned in the beginning of this research, one of the outcomes is to obtain a comparative study between the main objective approach and past existing software test oracle methods generated in the exact same experimentation and generated with the exact data. This will provide us with a high-level impact and understanding of where support vector machine applied as an automated test oracle stands with the rest of the pattern recognizers. However, take note that the outcome of this comparison does not dictate that

one specific testing method i.e. one pattern recognizer is better than the other. Based on the research methodology we understand that each pattern recognizers have different purposes, strong suites and weak points.

To conduct this comparative study, Artificial Neural Networks is chosen as the benchmark to compare against the main objected pattern recognizer SVM+K-NN. As we understand ANN is one the most widely studied and used pattern recognizer in software testing when implicated to software behavior. In Chapter 2 we explain how ANN functions as an automated test oracle and show the results based on past research. Therefore, to obtain an accurate comparative result, we used the exact same test data and training data that was used in the experiment for the support vector machine. Everything applied in an ANN structured framework using cleansed data.

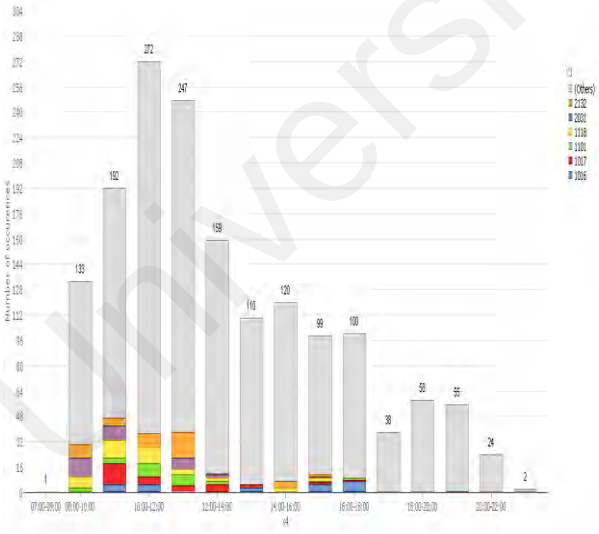

Figure 5.1 Frequency Chart of the Input Data

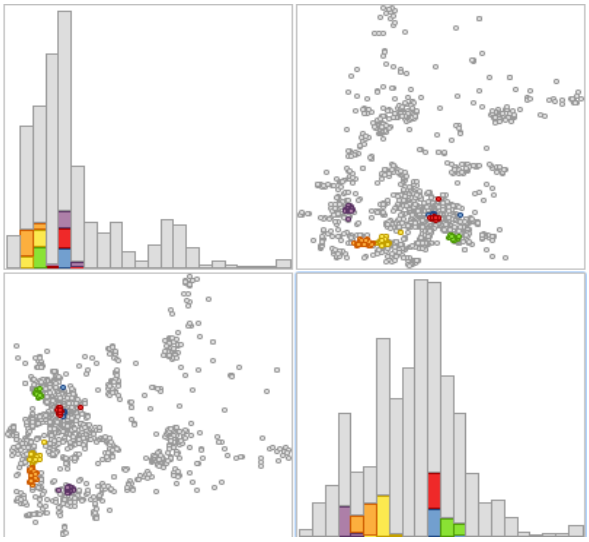

Figure 5.2 Scatter Matric of the Input Data

The figures above show the histogram frequency chart of the input data generated in an ANN framework. The diagram shows the number of occurrences i.e. exact matches cross checking with the geocodes, as geocodes are the key identifier of the experimentation. The

positive outcome of using ANN is that each data is processed and generated an occurrence and the ones that failed it will list as zero. Using this pattern recognizer, even the faulty data gets accounted for and hence this is where it differs from using SVM+K-NN. The result of using ANN as a pattern recognizer is as follows

Table 5.3 Experiment Using ANN

	ANN	**Remarks**
Master Data	45,757 records	The master data contains every necessary data such as: • Latitude • Longitude • Postal Codes • Actual time • Time windows Every input data that is inserted will be compared with this master data.
Input Data	1,610 records	The input data is months of worth of information based on a specified country. This input data also contains data such as: • Latitude • Longitude • Postal Codes • Actual time • Time windows
Run Time	120 seconds	ANN actually processes each and every outcome of the input data causing the run time to take much longer in comparison to the SVM,
Exact Match	212 records	
Potential Probability	0	ANN: Unable to calculate a probability for potential time windows.
False Match	1398	ANN: This got 1,398 false records out of 1,610 records. This leads to the lack of in accuracy, unable to calculate probability of potential time windows and lastly the input data used still contains faulty and inaccurate information.

Based on the table above, processing the experiment using ANN as the framework with the exact same master and input data used in the SVM+K-NN experimentation. The run time to generate the result was 2 minutes and this due to the fact that ANN actually executes each input as a whole (refer to Chapter 2 Figure 2.6-1), hence what occurs is that the input data may contain some errors and it causes a downtime for the framework to generate those errors. On the other hand, SVM has no problem when it comes run time due to its ability to

ignore faulty data and generate an output. On the exact match result, ANN generated 212 exact matches. It is well studied that ANN is known for its high accuracy rate and error reduction and in this case, is no different. By looking at the frequency chart and the scatter matrix above, the framework actually reads each and every input data and generates an outcome, resulting in no requirement for potential probability matches. Even for the data which the framework failed to find an exact match, it was also generated as a single entity. The reason for only 212 exact matches solely falls on the construction of the case study in other words we can conclude that ANN may not be as effective as SVM+K-NN when it comes to Time and Window predictions. ANN generated 1398 false matches due to the fact the pattern recognizer is unable to calculate potential matches hence reading the slightest difference such as 09:00 – 11:00 and 10:00 – 12:00 results in a failed matched. SVM+K-NN is able to read that whole and generate a time window prediction based on the amount of times the courier arrived to deliver a package (e.g. 09:45 – 10:15) as 09:00 – 11:00.

5.3 The Comparative Conclusion

The conducted experiments and their results show clearly that the use of any pattern recognizers maybe effective but are solely depending on the type of the testing method required.

In a brief note with a previous proposed ANN solution by Shahamiri et al., (2012) as mentioned in Chapter 2, the related work used a different case study based on a web-based car insurance application. The proposed solution used 9121 records of input data and has achieved in the highest threshold a 91.83% accuracy rate for the Single-Network Oracle and a 98.26% accuracy rate on a Multi-Network oracle, both results were using the ANN

approach. However due to the inability to obtain access to those data, a different case study based on courier company time window prediction was applied for the current proposed SVM+KNN solution as well as the ANN approach to obtain a comparative result. Using SVM+K-NN as an automated test oracle as obtained an accuracy rate of 52% and a performance result of 1 minute and 12ms to generate the necessary results. Whereas using the ANN approach for the same case study and data obtained an accuracy rate of 15% and a performance result of a 120s. For the specific test case the SVM+KNN was easier to learn, this is measured by the fact that SVM+KNN is able to retrieve a higher value of exact matches then ANN as shown in the result tables above. Another advantage is that SVM+KNN contained a compatibility with multiple classifiers to choose from. The table below is a list of differences between the proposed model SVM and ANN model.

Table 5.4: Comparison Between SVM+KNN and ANN

Entity	Support Vector Machine & K-NN (SVM+K-NN)	Remark	Artificial Neural Networks (ANN)	Artificial Neural Networks (ANN)
Time	1m 12ms	Quicker processing and run time, due to ability to ignore faulty data.	120s	Take a longer time to process, as an entire input data is processed as a whole instead of an input data at a time. The time is extended further once encountered with fault in the data.
Accuracy	850 records exact match	High accuracy rate specifically during forecasting prediction due to the optimal separating hyper-plane, the SVM is able to read each data and compute which patterns fits.	212 records exact match	Low accuracy rate when it comes to forecasting predictions, however extremely accurate when conducted in classification approaches.
Patterns	SVM + KNN	Contains dual classifiers (e.g. SVM, KNN) to apply to any given scenario.	ANN	Functions solely has a pattern recognizer

Chapter 6

6 Conclusion

Automated testing is an important segment in the software development life cycle (SDLC). With the technological advances and companies investing more on software applications and development, automated testing methods are increasingly demanded. Manual testing although still being used widely today, is beginning to wear out gradually as companies and researchers realizing that its time consuming and costly. The existing research has increased and is concentrated in automated testing methods. With that being said, support vector machine applied as an automated test oracle is an entity of contribution to solving problems that pertains to testing methods. In this research, new approach was introduced that can take large amount of data and process them based on classification, regression clustering, time forecasting and many more. Based on the experimentations, an improvement has been made of nearly 40% increase compared to the ANN test oracle. This testing method will be able to assist testers in running automated testing based on their test cases, provide a more accurate outcome and reduce human error.

6.1 Research Objectives Revisited

6.1.1 Research Objective 1:

To ascertain the challenges in the existing software test oracle.

This objective has been achieved and explained in the literature review, the research method and comparative conclusion. The outcome is that existing oracles such as ANN and IFN has their purpose and strong suites as well as challenges such as IFN only being limited

to regression testing. For objective 1, there are three research questions that need to be considered as follows:

RQ1.What are the challenges with existing software test oracles?

In answering RQ1, the challenges regarding existing software test oracles always relates to the three main factors which are time, cost and accuracy.

6.1.2 Research Objective 2:

To identify a pattern recognition approach to address the challenges in an automated oracle in software testing.

This objective is achieved by the proposed method SVM as an automated test oracle. The proposed approach has able to address the challenges in an input/output relationship analysis by being able to generate in fast rate hence a large reduction in time consumption. For this objective 2 there is one research question that needs to be considered as follows:

RQ2. Why support vector machine?

RQ3. What are the advantages and disadvantages of applying support vector machine as a pattern recognizer?

RQ4. How does support vector machine address the challenges faced by existing pattern recognizers?

In answering to RQ2, discussed in chapter 2 Support Vector Machine (SVM) is one of the latest pattern recognizers in the area of pattern recognition that can improve the classification performance machine learning and data mining with the capability of performing both classification and regression. For RQ3 the largest advantages for applying support vector

machine as a patter recognizer is that SVM generalization error does not correlate with defected parts of the input data which provides a high performance when processing in high dimensional complex application. The disadvantage is the generation minor inaccurate data known as a non-liner problem which also effects the performance. However, patches have been applied to solve the problem and increase the accuracy and performance of the pattern recognizer. For RQ4 as discussed in chapter 5, support vector machine has a higher accuracy rate in comparison with ANN as well a higher performance due to the inability to process faulty data. SVM is also time effective, processing at a faster rate and generating the results.

6.1.3 Research Objective 3:

To measure the accuracy of the approach by the application of the developed tools in the selected case study, and the comparison with other existing prominent oracles.

This objective has been achieved in the experimentation phase using the case study based on the courier domain and a comparative study has been conducted with the proposed approach against ANN as an automated test oracle. For this objective 3, there is one research question that needs question that need to be considered as follows:

RQ5. How is the accuracy of the approach being measured?

RQ6. What is the case study being used?

RQ7. Against what test oracles is the approach compared with?

In answering to RQ5, the accuracy of the approach is being measured by using a java tool as mentioned in chapter 5, where the tool will output the amount of records have obtained an exact match, false match and a potential probability. For RQ6, the case study being used is based on a courier company provided data, focusing on time prediction window application,

where the courier company is able to predict the time of shipment delivery. For RQ7 as discussed in chapter 5, the approach is compared with software test oracles such as ANN and dual classifiers (SVM+KNN).

6.2 Research Contribution and Significance

The proposed SVM applied in an automated test oracle that tackles the main problems of accuracy, cost effectiveness and time consumption. The research discussed the need of software testing methods to better utilize pattern recognizers in the testing phases. Researching on different patter recognizers concluded in the use of classifiers to which not only shows that the testing method proposed is flexible but is also compatible. Using classifiers such as SVM and KNN assists the testing method to higher accuracy and faster run time when processing the data. Another core significance is in the experimentation which the result of time prediction windows. This is an attribute that courier companies and many other companies related with deliveries are trying to perfect. Using this method, based on the data, the proposed approach can access and make sense of the pattern of e.g. courier's everyday stops and is able to generate a report of a time prediction. This form of prediction is beneficial as courier companies can inform their customers with a more precise time window of when the delivery will arrive. This will create a domino effective of customer becoming more confident with the service they have chosen and for the companies increase in profits with the least amount of cost used to apply the method. In conclusion the proposed method has proven to be effective in addressing the objectives of this research.

6.3 Limitation of Study

Although the proposed approach has address the objectives and provides a solution to multiple problems, there are some limitations. As explained in section 2.11, using support vector machine contains some drawbacks and requires extra methods or classifiers, to help avoid disturbances during the testing process that will disturb the accuracy and performance of the pattern recognizer. These disturbances are non-linear relations that effect the standard SVM formulation hence concluding that SVM is not robust to this disturbance. However, with the use of patches to programs and proposed solutions is able to learn the non-linear relations with a linear machine.

6.4 Future Work

Automated testing is still accounted for as new and contains a vast room of improvement and that is including in the use of SVM as classifier. Even when conducting this research many ideas were thought of and brought up but were not applied simply because there require their own separate efforts or because they are not within the scope of the research. In addition to that the most pressing effort currently is to create a proper tool as software or ideally an IDE plugin that is able to conduct training data and processing the proposed testing framework. More so the tool is able to provide more detail of a generated report with histograms, scatter matric plots, pie charts, automated line graphs for clustering and many more.

References

Agarwal, D., Tamir, D. E., Last, M., & Kandel, A. (2012). A comparative study of artificial neural networks and info-fuzzy networks as automated oracles in software testing. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, *42*(5), 1183–1193. https://doi.org/10.1109/TSMCA.2012.2183590

Bhatia, N., & Author, C. (2010). Survey of Nearest Neighbor Techniques. *IJCSIS) International Journal of Computer Science and Information Security*, *8*(2), 302–305.

Chih-Wei Hsu, Chih-Chung Chang, and C.-J. L. (2008). A Practical Guide to Support Vector Classification. *BJU International*, *101*(1), 1396–400. https://doi.org/10.1177/02632760022050997

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Keller, J. M., & Gray, M. R. (1985). A Fuzzy K-Nearest Neighbor Algorithm. *IEEE Transactions on Systems, Man and Cybernetics*, *SMC-15*(4), 580–585. https://doi.org/10.1109/TSMC.1985.6313426

Leif E. Peterson. (2009). K-nearest neighbor. *Scholarpedia*.

Li, X., & Shu, L. (2008). Fuzzy Theory Based Support Vector Machine Classifier. *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, *1*, 600–604. https://doi.org/10.1109/FSKD.2008.440

Schroeder, P. J., Faherty, P., & Korel, B. (2002). Generating expected results for automated black-box testing. *Proceedings - ASE 2002: 17th IEEE International Conference on Automated Software Engineering*, 139–148. https://doi.org/10.1109/ASE.2002.1115005

Shahamiri, S. R., Kadir, W. M. N. W., & Ibrahim, S. (2010). A single-network ANN-based oracle to verify logical software modules. *ICSTE 2010 - 2010 2nd International Conference on Software Technology and Engineering, Proceedings*, *2*, 272–276. https://doi.org/10.1109/ICSTE.2010.5608808

Shahamiri, S. R., Kadir, W. M. N. W., Ibrahim, S., & Hashim, S. Z. M. (2011). An automated framework for software test oracle. *Information and Software Technology*, *53*(7), 774–788. https://doi.org/10.1016/j.infsof.2011.02.006

Shahamiri, S. R., Kadir, W. M. N. W., & Mohd-Hashim, S. Z. (2009). A comparative study on automated software test oracle methods. *4th International Conference on Software Engineering Advances, ICSEA 2009, Includes SEDES 2009: Simposio Para Estudantes de Doutoramento Em Engenharia de Software*, 140–145. https://doi.org/10.1109/ICSEA.2009.29

Shahamiri, S. R., Wan-Kadir, W. M. N., Ibrahim, S., & Hashim, S. Z. M. (2012). Artificial Neural Networks as multi-networks automated test oracle. *Automated Software Engineering*, *19*(3), 303–334. https://doi.org/10.1007/s10515-011-0094-z

the Art of Software Testing James L . Dalley Eastman Kodak Company , Dayton Operations. (1991), 757–760.

Vanmali, M., Last, M., & Kandel, A. (2002). Using a Neural Network in the Software Testing Process. *International Journal*, *17*, 45–62.

Whittaker, J. a. (2000). What is software testing? And why is it so hard? *IEEE Software*, *17*(1), 70–79.

Whittaker, J. A. (2000). What is software testing? And why is it so hard? *IEEE Software*,

17(1), 70–79. https://doi.org/10.1109/52.819971

Xue, H., Chen, S., & Yang, Q. (2011). Structural regularized support vector machine: A framework for structural large margin classifier. *IEEE Transactions on Neural Networks*, *22*(4), 573–587. https://doi.org/10.1109/TNN.2011.2108315

Yousif, M. E. (2015). Test Oracle based on ANN and IFN , A Comparative Study.