IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-1, Jan 2019 E-ISSN: 2347-2693

Single and Multi Network ANNs as Test Oracles – A Comparison

J. Mary Catherine[1*], S. Djodilatchoumy[2]

[1]Periyar University, Salem (CTTE College for Women, Chennai -11).
[2]Department of Computer Science, Pachaiyappa's College, Chennai- 30

[*]*Corresponding Author: catherine_vinod@yahoo.co.in, Tel.: +91 9543326499*

Available online at: www.ijcseonline.org

Accepted: 26/Jan/2019, Published: 31/Jan/2019

Abstract— Software testing, which once was a distinct phase in software development life cycle, has now become a parallel activity. Many researchers in the past have attributed the failure of software to the lack of adequate testing. Software testing involves checking whether the actual outputs generated by the SUT matches the expected outputs. Test cases are written and executed and the results are compared with the help of a test oracle. A Test Oracle is a mechanism to determine whether a test has passed or failed. The process of finding a reliable test oracle is called the oracle problem. Software test automation has been a hot area of research for more than a decade. But, the work in the area of test oracle automation is minimal. Some of these researches have proposed solutions for test oracle automation using machine learning algorithms like Genetic Algorithms (GA) and Artificial Neural Networks (ANN). In this paper, we present a brief review and comparative analysis of the use of single-network and multi network ANNs as test oracles.

Keywords— Software Testing, Artificial Neural Networks, Test Oracles, Machine Learning, SDLC.

I. INTRODUCTION

Software Testing is one of the key areas of software development life cycle (SDLC). Once a distinct phase, it has now become an inevitable part of SDLC. The earlier the faults are discovered, the lesser the cost of repairing it. Many researchers in the past have attributed the failure of software to the lack of adequate testing. A better relationship between testing and requirements was laid down as a pre requisite for better-quality software [1], leading to establishment of test methods [2].

Software testing is done at various levels namely unit testing, regression testing, module testing, integration testing, system testing and acceptance testing. Broadly, these levels are classified into two types of approaches: White-box and Black-box testing. White-box testing is done mostly by the programmer with the code in hand. It is also referred to as Structural Testing. Black-box or functional testing considers the software as a black box with hidden implementation. The inputs are given to the software under test and actual outputs are compared with expected outputs.

The mechanism used to compare the expected outputs with the actual outputs to determine whether the test has passed or failed is called a test oracle. The process of finding correct and reliable expected outputs is called oracle problem [4]. The Structure of an automated test oracle is depicted below in Figure 1.

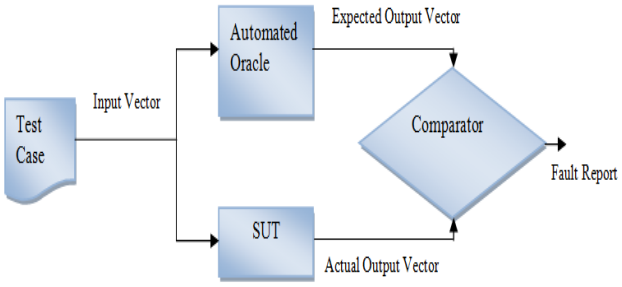

Figure 1: Structure of a Test Oracle

Test Oracle automation requires mapping the input domain to the output domain automatically and compare actual and expected results in order to verify the correctness of the SUT behavior. According to [5], it is difficult and expensive to map the input to the output domain manually, hence an automated test oracle's first task is to automate input-output mapping. Also the comparator needs to define thresholds instead of directly comparing the actual and expected outputs in order to be tolerant.

In this paper, we present a brief survey on the existing work done in the area of test oracle automation; and a comparative analysis of Single network and Multi network ANNs as Test Oracles.

The rest of the paper is organized into sections. Section II gives a brief account of the work done by various researchers

in this area. Section III gives a brief introduction to Artificial Neural Networks (ANN) as a Test Oracle. Section IV provides the details of the methodology and case study used in the experiment. Section V describes the result and Section VI provides the conclusion and future work.

II. RELATED WORK

[7] proposed a partial test oracle to prove the correctness of observed results in XML data. The SUT considered was a query program which takes XML document as input acquired from an XML repository of any form and produces XML data as a result. The oracle automatically provides a response about the accuracy of the module being tested with a certain degree of accuracy at feasible cost.

[8] exploited artificial intelligence and pattern recognition techniques to develop a pattern recognition based oracle, in which, the unit emulates human tester's knowledge and decision making process. The oracle imitates an accurate version of the module tested and makes use of a distance measure to check the distance between the tested module output patterns and output patterns of oracles. The oracle problem exist when either the oracle does exist or not but very expensive to be used.

[9] used metamorphic relations, properties of the SUT represented in form of relations amongst the inputs and the outputs of multiple executions, in order to aid validate the perfection of a program. This approach was used to lessen the oracle problem in numerous systems and to improve numerous software examination and testing methods. The authors witnessed that identification of an adequate amount of suitable metamorphic relations for testing by naïve testers was feasible with minimum training. Besides, the approach is cost-effectiveness and could be enhanced using more diverse metamorphic relations.

A novel functional testing approach was attempted in [10] to verify the test oracles. The expected execution output for a given application is produced and verified by the oracle whether the SUT behaved correctly or not and issues a pass or fail verdict. [11] proposed a model based oracle generation method for unit testing. A fault model was developed based on features of the main components to record the kinds of defects that may be faced, and describe how to generate automatically a passive, partial oracle from the agent design models. Khoshgoftar et al. [12] proposed a method to employ ANNs predicting the number of faults in SUT based on the software metrics. ANNs were used as a regression test oracle by Vanmali, Last and Kandel [13]. They modeled an ANN to simulate the software behavior using the previous version of the SUT, and applied this model to regression test unchanged software functionalities. Using the previous version of the SUT, expected outputs were generated and the I/O pairs were employed to train the ANN. Aggarwal et al. applied the same approach to solve the triangle classification problem [14].

The approaches mentioned above modeled and tested discrete functions. Mao et al. formulated ANNs as test oracles to test continuous functions [15]. Consider the continuous function $Y=F(x)$, where x is the software input vector, y is the corresponding output vector and F is the software behavior. The function F was modeled and expected outputs were generated using a trained ANN. Instead of Perceptron Feed-Forward neural networks that applied in all of the above researches, Lu and Mao [16] used *RBF neural networks* to provide automated oracles and test a small mathematic continuous function.

S. R. Shahamiri, W. K. Wan M. N. and S. Ibrahim have proposed single-network and multi-network ANN based test oracles in [18] and [19] respectively whose comparative analysis is discussed below.

III. ANN AS A TEST ORACLE

In this section, we discuss in brief about artificial Neural Networks (ANN) and its applications as a test oracle.

3.1. Artificial Neural Networks

Neural Networks are network structures comprised of some correlated elements called *neurons*, each one has input(s) and output(s), and they perform a simple local *add* operation. Each neuron input has its corresponding *weight*, which it acts as the neuron memory. The neurons have their *Bias*, *Weights* and *Activation Functions* which can be adjusted to learn the hidden knowledge being modelled through a process called the *Training Process*. The accuracy of the ANN depends on how well the network structure is defined and the training process is done. *Learning Rate* is one of the training parameters that show how fast the ANN learns and how effective the training is. It can be in range 0- 1.

Multilayer Perceptron networks are one of the most popular types of ANNs to solve the non-linearity. They are multi-layered networks with no limitation to choose the number of neurons, and they must have an i*nput layer*, one or more *hidden layers* (or middle layers) and one *output layer*. The structure is depicted in Figure 2 below.

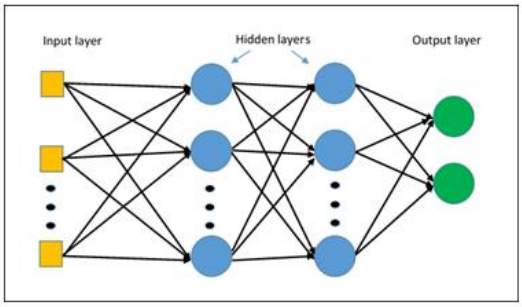

Figure 2: Structure of a ANN

The general model of Perceptron networks is *Feed-Forward* with *Back-Propagation* training procedure, which feed-forward are networks with inputs of the first layer connected and propagated to middle layers, the middle layers to the final layer, and the final layer to the output layer. In back-propagation procedure, after results of the network are generated, the parameters of the last layer to the first layer will be corrected in order to decrease the network misclassification error. The misclassification error can be presented by Mean Square Error (*MSE*), which it is the squared difference between the training sample outputs and the results generated by the network. [6]

3.2. Single Network Test Oracle (SNO)

A Single-Networks Oracle, models the entire I/O domain using only one ANN. In particular, the ANN must learn the required functionalities to generate all outputs itself. For example, if the SUT has 5 outputs in its output domain, then the entire 5 outputs has to be modelled using a single ANN. The structure of the SNO is given below. As the number of outputs increases, the complexity of the training process also increases. As complex software applications may require a huge training dataset, the practicality of Single-Network Oracles to find faults may be reduced.

3.3. Multi Network Test Oracle (MNO)

A Multi-Networks Oracle is made up of several Single-Network Oracles in parallel, one for each output domain. In our case study, we have four output items, thus we need four ANNs to create the Multi-Networks Oracle. Since the complexity of the SUT is distributed among several ANNs, each ANN has less to learn so it eases the training process; thus, it is easier for the ANNs to converge on the training data. Each ANN of the multi –network oracle must be trained separately using the same input vectors but only the output to be generated by the ANN. Figure 3 below depicts the Single network and Multi-Network Test Oracle structure.

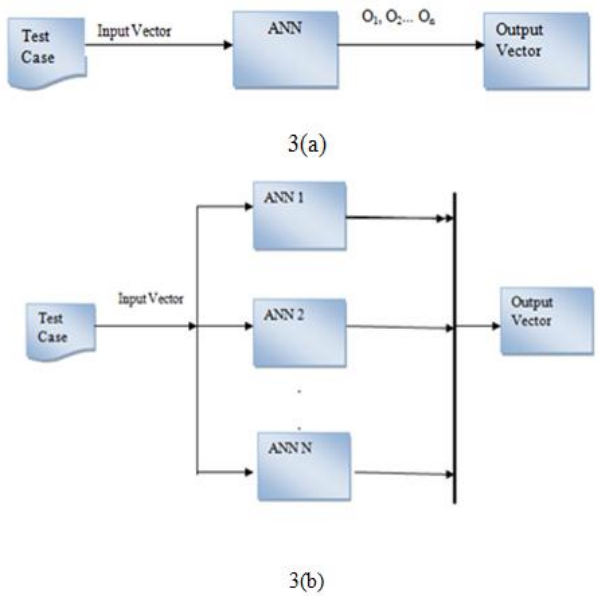

Figure 3: (a) Single network and (b) Multi network Test Oracle structure

3.4. Training the Test Oracle

Before training the all non – numeric inputs and output in the dataset are normalized to numeric values. Binary values are represented with 0 and 1. Continuous numeric values are scaled to values between 0 and 1. The ANN is trained with input-output pairs from the dataset. In order to increase the quality of the network, the ANN results are compared with correct expected results provided by the training samples. The network error (MSE) can be measured by calculating the distances between the expected results and the network-generated results. To achieve an adequate error rate, the networks parameters (neurons and biases weights mentioned in previous section) are adjusted by back-propagating the error data to the network. Then, the whole process is repeated and continued until adequate error rate being obtained.

In the case of single-network oracle the entire output domain is generated by a single ANN, while in case of multi-network oracle, each ANN generates its associated output for the training data.

3.5. Using ANN as a Logical Test Oracle

After the ANN trained correctly the ANNs can be used as a Single-Network Oracle (S-NO) and Multi-network Oracle (M-NO) respectively. The process is as follows: Given the test cases, the logical module under test is executed using the input vector provided by the test case; meanwhile, the oracle is given by the same inputs. Actual outputs, which are supposed to be evaluated, are generated by the SUT and the expected outputs by the ANN. In case of S-NO, expected outputs are generated by a single ANN. For M-NO, each of the expected outputs are generated by the ANN which is trained for that output. A comparison process is required to compare these outputs, and any difference is reported as a possible fault, considering the thresholds that define the comparison tolerance.

IV. EXPERIMENT

The above SNO and MNO were evaluated using our case study on sample competitive exam registration portal. The registration page was implemented using ASP.NET with VB using MS-Visual Studio package. The input domain consists of six parameters and output domain has two parameters. The input domain and the equivalence class partitions are given in Table 1. Equivalence classes partitioning, which is a test case reduction technique, was used to generate optimum number of test data. As seen from the table, there are 360 equivalence classes for the input domain. The test cases have been carefully chosen to cover all the 360 combinations. The test data was generated using online test data generation tool. The output domain consists of two outputs as below:

1. Output 1: Allowed to register – Yes/No
2. Output 2: Fees discount – Yes/No

Table 1: Input Domain of the case study

Input	Equivalence Classes	Number of Equivalence Classes
Nationality (Indian, NRI, OCI, PIO, Foreign National)	Indian NRI OCI PIO Foreign National	5
Age (18-25)	<18 18-25 >25	3
Category (Reserved, Unreserved)	Reserved Unreserved	2
Qualifying Marks (40-100)	<40 40-59 60-100 >100	4
Attempt number (Maximum 3)	<3 3 >3	3
Total number of Equivalence Classes: 5 X 3 X 2 X 4 X 3 =360		

The SNO and MNO were implemented as Multi-Layer Perceptron Neural Networks (MLP-NN) using Weka, an open source tool. The S-NO was implemented as a single MLP-NN, while the M-NO was implemented as two MLP-NNs, one for each output. The quality of the oracle was verified against accuracy, precision, misclassification error and recall by evaluating the model using mutation testing. Tables 2 and 3 show the implementation details of the ANNs.

Table 2: MLP specifications for SNO

Parameters	Values
Input Neurons	6
Hidden Neurons	40
Output Neurons	2
Learning rate	0.01
Training Cycles	1000
Mean Squared error (MSE)	Output 1 : 0.01764 Output 2 : 0.00081 Total MSE : 000567

Table 3: MLP specifications for MNO

Parameters	ANN 1 Values	ANN 2 Values
Input Neurons	6	6
Hidden Neurons	30	30
Output Neurons	1	1
Learning rate	0.01	0.01
Training Cycles	1000	1000
Mean Squared error (MSE)	0.0073	0.00001
	Total 0.00021	

Two versions of each case study were generated. The Golden Version, which was a complete fault free implementation of the case study the generated correct, expected results. The Mutated Version was injected with common programming mistakes. The inputs were passed simultaneously to the Golden and Mutated versions of the SUT and the ANN oracle.

After the test cases are executed, the results of the golden version, the ANN and the case study were compared with each other. The comparison process measured the distance between the golden version result and the oracle result, and between the oracle and the mutated results. The golden version produces correct expected outputs, the oracle produces oracle outputs, and the case study generates mutated outputs. Then, the process compares the distances by a defined threshold that determines the comparison tolerance and reports one of the below possibilities:

- True Positive, implies that there is actually neither any fault in the mutated version or the oracle. True positive represents the successful test cases.
- True Negative, the oracle results are correct and the oracle correctly finds a fault in the mutated version.
- False Positive, occurs when both the oracle and mutated version produced the same incorrect results, which means it has missed a fault.
- False Negative, implies the mutated and the expected results are the same, but they are different from the oracle results. Thus, the comparator reports a faulty oracle.

3000 random test cases were executed to verify the oracles. The thresholds for the two binary outputs were not adjusted because their MSEs and absolute errors are tiny (almost zero). In addition, the distance between the binary output values (true or one, and false or zero) is large enough. The thresholds for the first output from ANN1, and the second output from ANN2 were chosen to be 0.08 and 0.015 respectively in order to increase the precision of the oracle.

V. RESULTS

The above ANNs were executed and the results are summarised in Table 4. Table 4: Comparison of SNO and MNO

Parameters	SNO	MNO
Average Threshold	0.06	0.06
Total Comparisons	6000	6000
Number of injected faults	2064	2156
Total Absolute Error	0.027	0.020
True Positive	3852	3821
True Negative	1909	2063
False Positive	96	84
False Negative	143	32
Misclassification Rate	3.98	1.93
Accuracy (%)	**96.02**	**98.07**

The total absolute error obtained in case of S-NO and M-NO were 0.027 and 0.020 respectively. The misclassification rate is calculated as the ratio of the sum of false positives and false negatives to the total comparisons made. The misclassification rate obtained in case of S-NO and M-NO were 3.98% and 1.93% respectively. The accuracy obtained in case of S-NO and M-NO were 96.02% and 98.07% respectively.

VI. CONCLUSIONS AND FUTURE WORK

The results clearly indicate that Multi-Network Oracles show a slight improvement to the Single-Network Oracles. Though there is a marginal increase in accuracy of MNO over SNO, the implementation of MNO is relatively easy as compared to SNO. As the complexity of the software increases in terms of number of inputs and outputs, MNO provides a simpler structure to implement.

Our future work will be to propose a hybrid test oracle using genetic algorithms and ANNs especially in the area of GUI test automation.

REFERENCES

[1] Suresh Jat., Pradeep Sharma., "*Analysis of Different Software Testing Techniques"*. International Journal of Scientific Research in Computer Science and Engineering Vol.5, Issue.2, pp.77-80, April 2017.

[2] Chandraprakash Patidar., "*A Report on Latest Software Testing Techniques and Tools"*. International Journal of Scientific Research in Computer Science and Engineering Vol.1, Issue.4, pp.50-52, Dec 2016.

[3] [J. A. Whittaker, "*What is software testing? And why is it so hard?"* IEEE Software, Vol. 17, pp. 70-79, 2000.

[4] P. Ammann, and J. Offutt, "*Introduction To Software Testing*". Camberidge University Press, 2008.

[5] Q. Xie and A. M. Memon, "*Designing and comparing automated test oracles for GUI-based software applications*", ACM Transactions on Software Engineering and Methodology, Vol. 16, pp. 4, 2007.

[6] S. R. Shahamiri, W. K. Wan M. N. and Z. M. H. Siti. "*A Comparative Study on Automated Software Test Oracle Methods*" Proceedings of International Conference on Software Engineering Advances (ICSEA'09), IEEE Press, Sep 2009.

[7] R. J. Schalkoff, "*Artificial Neural Networks"*. McGraw-Hill, 1997. [15] M. B. Menhaj, "*Basics of Neural Networks*". Amirkabir Technology University, 2001.

[8] Kim-Park, D.S.; de la Riva, C.; Tuya, J., "*A Partial Test Oracle for XML Query Testin*," Testing: Academic and Industrial Conference - Practice and Research Techniques, 2009. TAIC PART '09. , vol., no., pp.13,20, 4-6 Sept. 2009.

[9] Gondra, "*Applying machine learning to software fault proneness prediction*" J. Syst. Softw., vol. 81, no. 2, pp. 186–195, Feb. 2008.

[10] Huai Liu; Fei-Ching Kuo; Towey, D.; Tsong Yueh Chen, "*How Effectively Does Metamorphic Testing Alleviate the Oracle Problem?"* Software Engineering, IEEE Transactions on , vol.40, no.1, pp.4,22, Jan. 2014.

[11] Monisha, T.R.; Chamundeswari, A., "*Automatic verification of test oracles in functional testing*" Computing, Communications and Networking Technologies (ICCCNT),2013 Fourth International Conference on , vol., no., pp.1,4, 4-6 July 2013.

[12] Padgham, L.; Zhiyong Zhang; Thangarajah, J.; Miller, T., "*Model-Based Test Oracle Generation for Automated Unit Testing of Agent Systems*" Software Engineering, IEEE Transactions on , vol.39, no.9, pp.1230,1244, Sept.2013.

[13] T. M. Khoshgoftaar, A. S. Pandya, H. B. and More, "*A neural network approach for predicting software development faults*" Proc. Third IEEE International Symposium on Software Reliability Engineering , pp. 83-89, 1992.

[14] M. Vanmali, M. Last and A. Kandel,"*Using a neural network in the software testing process*" International Journal of Intelligent Systems, Vol. 17, pp. 45-62, 2002.

[15] K. K. Aggarwal , Y. Singh, A. Kaur and O. P. Sangwan, "*A Neural Net based Approach To Test Oracle*" ACM Software Engineering Notes, 2004.

[16] Y. Mao, F. Boqin, Z. Li and L. Yao, "*Neural networks based automated test oracle for software testing*", in Neural Information Processing, Vol. 4234, Springer Verlag, pp. 498-507, 2006

[17] Y. Lu and M. Ye, "*Oracle model based on RBF neural networks for automated software testing*" Information Technology Journal, Vol 7, 2007, pp. 469-474.

[18] S. R. Shahamiri, W. K. Wan M. N. and S. Ibrahim. "*Artificial neural networks as multi-networks automated test oracle*" Automated Software Engineering (2012) 19:303–334.

[19] S. R. Shahamiri, W. K. Wan M. N. and S. Ibrahim. "*A Single-Network ANN-Based Oracle to Verify Logical Software Modules*" Proc. 2010 2nd International Conference on Software Technology and Engineering(ICSTE)

Authors Profile

Mrs. J. Mary Catherine pursed Bachelor of Science and Master of Science from Osmania University in year 2001 and 2003 respectively. She is currently pursuing Ph.D in Periyar University, Salem, Tamilnadu and currently working as Assistant Professor in Department of Computer Science, Chevalier T.Thomas Elizabeth College fo Women, Chennai, Tamilnadu, since 2008. Her main research work focuses on Software Testing, Neural Networks and Algorithms. He has 10 years of teaching experience and 4 years of Research Experience.

Dr.S.Djodilatchoumy pursed Bachelor of Engineering(Computer Science and Engineering) from Anna University, Chennai, Tamilnadu in 1988, Master of Technology from PunjabUniversity in 2003 and Ph.D. from Mother Terasa University in 2011.Currently working as Head of the Department of Computer Science, Pachaiyappa's College,Chennai, India since 2008. She has published more than 15 research papers in international journals/Conferences and it's also available in online. Her main research work focuses on Neural Networks in Medical diagnosis using Spectral data. She has 7+ years of research experience.

