Performing Software Test Oracle Based on Deep Neural Network with Fuzzy Inference System

Amin Karimi Monsefi[1], Behzad Zakeri[2(✉)], Sanaz Samsam[3], and Morteza Khashehchi[2]

[1] Shahid Beheshti University, Tehran, Iran
a_karimimonsefi@sbu.ac.ir
[2] University of Tehran, Tehran, Iran
{Behzad.Zakeri,m.khashehchi}@ut.ac.ir
[3] Sharif University of Technology, Tehran, Iran
samsam.sanaz@ae.sharif.ir

Abstract. One of the challenging issues in software designing is testing the product in different condition. Various software Oracles had suggested in the literature, and the aim of all of them is minimizing the time and cost of the testing process. Software test Oracles have designed to do this job automatically with as less as possible human contribution. In this work, a novel Oracle based on deep learning and fuzzy inference system introduced. For this purpose, by the utility of Takagi-Sugeno-Kang fuzzy inference, the output of software mapped to the fuzzy space, and the deep neural network has trained by this data. Finally, data has remapped to the primary form and used as the competitor stage input. To validate the performance of the Oracle, four different models have chosen to assess the Oracle enforcement, and after training the Oracle by the correct output of applications, source codes have changed manually, and the efficiency of the Oracle monitored. Several measures have been applied to evaluate the accuracy of the test Oracle, and it is observed that in most cases Oracle correctly could detect the correct and false results. Finally, designing Oracles requires several preliminaries and in this work we only focus on the architecture of the system.

Keywords: Software testing · Deep learning · Oracle problem · TSK fuzzy inference system

1 Introduction

Software testing is a vital part of software technology to test the quality and reliability of computer programs under various conditions. Since human-based software testing requires too much time and energy resources, complete testing of software in limited time is impossible. For this purpose, automatic testers have been suggested to decrease the cost of this process [3]. Oracles have widespread

L. Grandinetti et al. (Eds.): TopHPC 2019, CCIS 891, pp. 406–417, 2019.
https://doi.org/10.1007/978-3-030-33495-6_31

applications in testing different kinds of software, such as web search engines, embedded software and video games [5,6,10].

A test oracle is a mechanism to assess the performance of the software. It is a valid source to study the operation of the under-test software [25]. It also generates different test cases considering software specification, and evaluate the actual behaviour of software [16]. Test oracles are useful in forming the automated software testing platform.

To assess the correctness of the under-test software, after the result generation process, Oracle should compare the generated results by software with correct results. It is conventional to call produced results using under-test software as *actual outputs*, and the correct results which are utilized to verify actual output known as *expected outputs* [25]. To check the actual outputs, Oracle has to find suitable expected outputs; this process of finding proper expected outputs mostly known as oracle problem [1].

The function of an arbitrary test oracle to evaluate test results starts with the generation of expected outputs. The second step is executing the test cases. In the next stage, the initial domain should be mapped to the expected outputs, and corresponding actual outputs for each expected output should be specified. Finally, actual outputs compare with expected outputs to find out whether the behaviour of the software is accurate or not [20]. Although test oracles designed to tackle all of these stages, in most studies oracles are used to test the execute cases only.

The aim of test oracles is tackling the software testing process with as less as the human contribution. However, this process faces several challenges in each automation activities steps [18,20]. The preliminary challenge in software testing is automatic data generation, and Oracle should provide proper output results automatically. Another challenge is that Oracle should provide expected results of the corresponding software inputs, and this is impossible without correct automatic mapping between input domain and output domain. The last issue in Oracle function is making the decision about which actual outputs are acceptable and which results should be considered as a fault [19].

In this study, we focus on forming a precise oracle by taking advantage of deep learning and fuzzy logic. For this purpose, by using Takagi-Sugeno-Kang (TSK) structure, a sufficient number of software inputs and their corresponding expected outputs have been encoded to the fuzzy space [21,23]. By utility of these data, a deep neural network has trained. This network with cooperation with fuzzy encoder-decoder performs our target Oracle.

2 Backgrounds

In the year of 1965 fuzzy logic has been suggested by prof. Lotfi A. Zadeh to cover the inabilities of the classical logic [27]. In this framework instead of using 1's and 0's to evaluate the value of parameters, variables can have a value in the spectrum domain between 0 and 1 [26]. To assign a spectrum value to each variable, the Fuzzy Inference System (FIS) is using for mapping the value of data

from binary space to fuzzy space. There are several FIS methods for mapping to fuzzy space, however, in action, only two FIS techniques are favourable which are TSK and Mamdani fuzzy models [13,17].

There are several differences between the TSK and Mamdani fuzzy structures. Computational cost is the main difference between these two structures. While Mamdani fuzzy model by computing the whole membership function, TSK model use simple formulas to computing the output. This feature of TSK structure makes it more useful FIS technique rather than the Mamdani model.

On the other hand, deep learning as a multilayer neural network with self-relying ability in feature extraction plays a crucial role in the prediction of systems' behaviour [11]. The range of deep learning applications covers a vast area of engineering and scientific topics, such as object detection, image classification and web search engines [4,7,8]. A deep neural network by the utility of several inputs and their corresponding output get trained and after the training procedure, can precisely predict the results of the system for arbitrary inputs.

Adaptive-network-based fuzzy inference system (ANFIS) is the most famous form of using the fuzzy logic in the neural networks. ANFIS is a kind of artificial neural network based on the TSK fuzzy inference system. This network has suggested by prof. Roger Jang in the early 1990s [9]. Since ANFIS uses the advantages of neural networks and fuzzy logic simultaneously, it has the capability of dealing with a wide range of problems like the prediction of nonlinear functions.

3 Methodology

In this study, with the integration of the advantages of deep learning and fuzzy inference system, it is tried to perform more efficient and accurate software test Oracle. Figure 1 shows the structure of the Oracle with containing two main stages which are Fuzzy encoder-decoder and deep neural network. The former stage by the utility of TSK structure maps the *I/O* of the under-test software to 0 to 1 and reverse it at the end of the Oracle structure. The latter stage uses these data to train and test of the deep neural network as the core of the Oracle. In the following context, each step will be discussed in detail.

3.1 Fuzzy Encoder-Decoder

Various fuzzy inference system has been suggested in the literature, however, most of the fuzzy reasoning can be classified into three main categories. Amongst these categories, Takagi and Sugeno fuzzy inference which uses a linear combination of input variables plus a constant term, is more functional than other inferences [12,22]. To have a better understanding of the TSK mechanism, firstly this inference is discussed for two input and the single output, and then the formula for multi-input-single-output (MISO) TSK model will be posed.

A conventional form of TSK fuzzy structure with two inputs and single output express as:

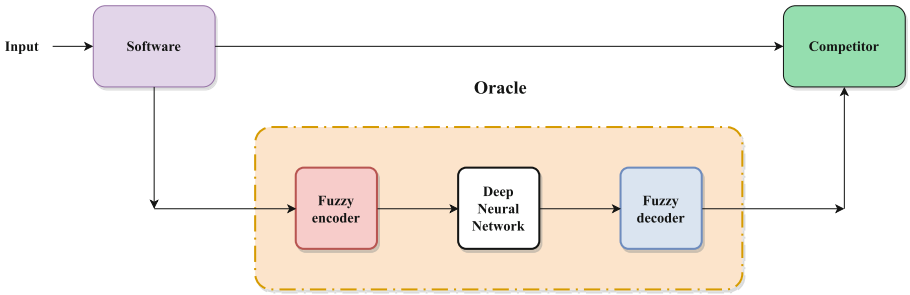


Fig. 1. Schematic Oracle structure

$$if\ x_1\ is\ A_i\ and\ x_2\ is\ B_i\ then\ y\ is\ y_i \tag{1}$$

Where in the Eq. 1, $A_i \in \{A^1, ..., A^{NA}\}$ and $B_i \in \{B^1, ..., B^{NB}\}$ represents the antecedent MF of the i^{th} rule that belongs to the input variables x_1 and x_2 respectively. The sets $\{A^1, ..., A^{NA}\}$ and $B_i \in \{B^1, ..., B^{NB}\}$ are pre-defined antecedent MFs. The i^{th} rule produces a partial output form which shown as:

$$y_i = f_i(x_1, x_2) \tag{2}$$

Where f_i are pre-defined functions in this study:

$$f_i(x_1, x_2) = r_i \tag{3}$$

And the $r_i = constant$, therefore characterizing a crisp consequent MF for the i^{th} rule. Aggregation the partial outputs of each rule, the output is given by:

$$f = \frac{w_1 y_1 + w_2 y_2}{w_1 + w_2} \tag{4}$$

Where $w_i = AND(\mu_{Ai}(x_1), \mu_{Bi}(x_2))$ is the weight of the i^{th} rule. The inference procedure has shown in the Fig. 2:

The mechanism of TSK inference system with two input and a single output for a better understanding of the operation of this model. However, it is noticeable that in this study all of the computations have conducted using MISO framework of TSK model. Since the schematic diagram of the MISO model and the governing equations are more complicated than the discussed model, in the following lines governing equation present briefly as:

$$R_i = if\ x_1\ is\ \widetilde{A_{11}}\ and(or)\ ...\ x_m\ is\ \widetilde{A_{1m}}\ then\ y = g(x_1, ...x_m) \tag{5}$$

Where m is the number of input variables, R is the fuzzy rule, $\widetilde{A_{ij}}$ is the fuzzy set corresponding to ith input variable for jth fuzzy rule and g_i is a function is defying as follow:

$$g(x_1, x_2, ..., x_m) = q + q_1 x_1 + ... + q_m x_m \tag{6}$$

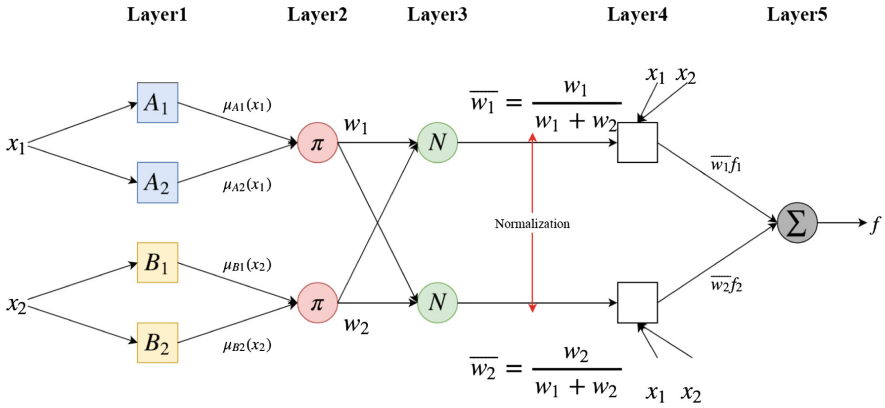


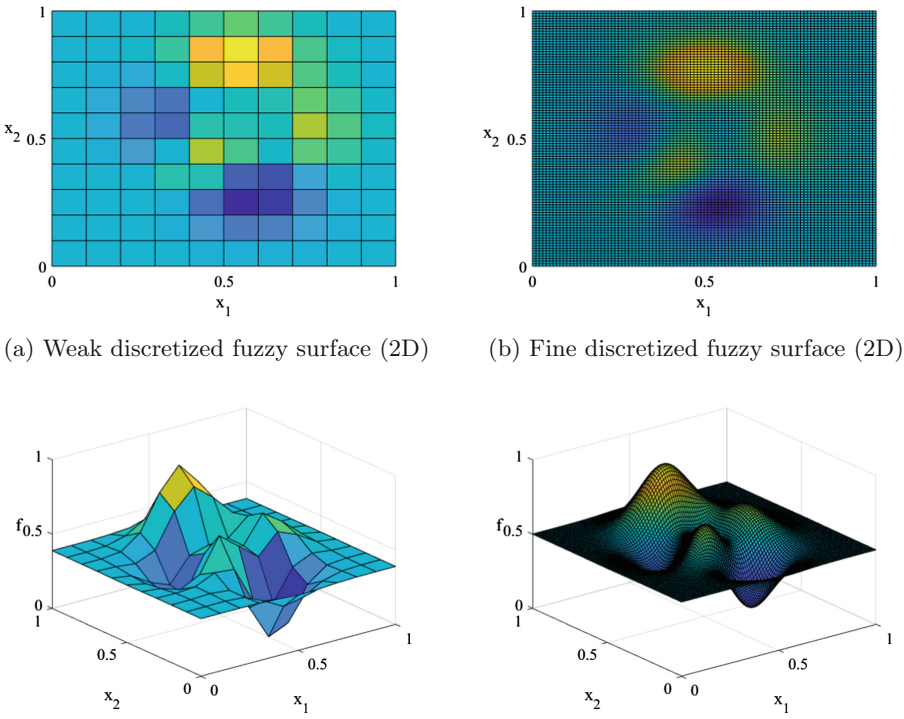
Fig. 2. ANFIS structure with two input variable

And finally, the fuzzy system can be described as follow:

$$y = \frac{\sum_{i=1}^{R} g_j(.) T_{j=1}^{m_i} \mu_{ij}(x_j)}{\sum_{i=1}^{R} T_{j=1}^{m_i} \mu_{ij}(x_j)} \tag{7}$$

Where μ_{ij} is membership function for the A_{ij} fuzzy set, and $m_i (1 < m_i < m)$ and T the number of inputs to the fuzzy inference and T-norm operator respectively.

After mapping the inputs by using Eqs. 5, 6 and 7 to fuzzy space, an n-dimensional hyperspace form by input variables and their output value conducted by FIS. To have a better understanding of the relationship between fuzzy inputs and outputs, a hyper-surface fit to these points in a way that contains as more as possible of points. Since digital computers and especially deep learning algorithm cannot accept a continues parameter as input, this hyperspace has discretized into several smaller hyper-surfaces. Each hyper-surface contains some of the fuzzy variables, and obviously, by increasing the number of this sub-hyper-surfaces, the precision increase and it is easier to define which point belongs to which element. Although increasing the number of elements in this stage increase the precision, since this sub-hyper-spaces were used as the learning input parameters in the deep learning module, it could be causing the over-fitting in the learning procedure. For this reason, it is significant to find optimum size of these elements which have been done by several experiments in this study. Sample model of hyper-surface and its discretization, and increasing the precision by increasing the number of elements in 3D fuzzy space is shown in Fig. 3.



(a) Weak discretized fuzzy surface (2D)

(b) Fine discretized fuzzy surface (2D)

(c) Weak discretized fuzzy surface (3D)

(d) Fine discretized fuzzy surface (3D)

Fig. 3. Schismatic view of fuzzy space

3.2 Deep Learning

In this section, deep neural network as the core of the Oracle has described, and the governing equations of the deep learning algorithm have discussed.

The deep neural network architecture has divided into three main layers which are the input layer, hidden layer and output layer. Input layer receives the mapped data from the fuzzy inference and delivers them to the hidden layers. Hidden Layers by processing data try to fund the relation among data, and finally, the output layer reports the outputs of the user.

To study the governing equation for an arbitrary hidden layer, all the equations govern the lth layer have been driven as follow:

$$Z^{[l]} = W^{[l]} * A^{[l-1]} + B^{[l]} \tag{8}$$

$$A^{[l]} = g^{[l]}(Z^{[l]}) \tag{9}$$

Where in this network $W^{[l]}$ is the weight matrix which links the layer l to the layer $l-1$, and $w_{ij}^{[l]}$ is the weight between neuron i in layer l and neuron j in the layer $l-1$. Also $B^{[l]}$ is equal to bios vector for layer l.

In Eq. 9, $A^{[l]}$ is a the output matrix for layer l and $g^{[l]}$ is the activation function which employed in the calculations. The function $g^{[l]}$ is the utilized activation function in the *l'th* layer the activation function which used in the hidden layer is function of *LeakyRelu* which are explained as follows:

$$\text{LeakyRe}lu(x) = \begin{cases} x, & x > 0 \\ x * 0.001, & x \leq 0 \end{cases} \tag{10}$$

Firstly, the values of W will be randomly chosen between 0 to 1 and then the values of B will be considered zero. The main purpose of learning model method is to decrease the error function.

$$\min_{W,B} J(W, B) \tag{11}$$

$$J(W, B) = \frac{1}{m} \|Y' - Y\|_2^2 \tag{12}$$

Where Y' and Y are the estimated value and the expected value which is extracted from the case studies. To prevent the *Overfitting* in this part, the error function was changed as follows:

$$J(W, B) = \frac{1}{m} \|Y' - Y\|_2^2 + \frac{\lambda}{2m} \|W\|_2^2 \tag{13}$$

$$J(W, B) = \frac{1}{m} * (Y' - Y)^T (Y' - Y) + \frac{\lambda}{2m} W^T W \tag{14}$$

By conducting the discussed method, the deep neural network has been trained to predict the test cases results.

4 Results

In this part, it is tried to evaluate the performance of the discussed test oracle in action. Due to this, four different software has been used as our case studies which each one has specified features. By utility of these case studies, numerous test case were generated for training and testing the Oracle. After the training process, by using the generated test cases, the accuracy of the Oracle has assessed.

4.1 Case Studies

One of the significant issues in test Oracles is the ability to tackle with unknown source codes. For this purpose, four source codes have been chosen from www.codeforces.com. This web site is a host for competitive programming contests and contains more than 10 million source codes with various scopes.

Our case studies have written in Java and C++ languages, and the *I/Os* of these codes are in the numeric form. To the study of the complexity of these codes, two criteria were used. The first criterion is the number of the programs'

code lines. The other criterion is *Cyclomatic Complexity* (CC) which is a standard programming measure based on the independent paths of the programming system. This criterion makes it possible to measure the complexity of arbitrary source code [24]. Table 1 summarizes all of the test cases features as follow:

Table 1. Specifications of test cases

#	Name	# code's line	# Input parameter	CC	Code's Language
1	Polyline	202	6	37	Java
2	Really Big Number	277	2	56	C++
3	Magic Number	368	4	31	Java
4	Karen & Neighborhood	256	2	29	C++

4.2 Oracle Evaluation

To train the Oracle for each case study 30K test case have generated. Producing of these data for each case study by considering the boundary of the case and utility of adaptive random data generation method has been done [2]. Also, 5K data with the same method have generated for testing purposes. For labelling these data, they considered as the input of the source code of each case study. Since the generation of a large amount of data is quite hard, it had forced us to generate the under-qualified number of data, and for filling this gap, we train the Oracle model with the same data iteratively with at least 500 iterations for each case study.

To analysis, the error value of the Oracle model for each case study, *mean square error* (*MSE*) formula has employed. The mathematical description of *MSE* become as follow:

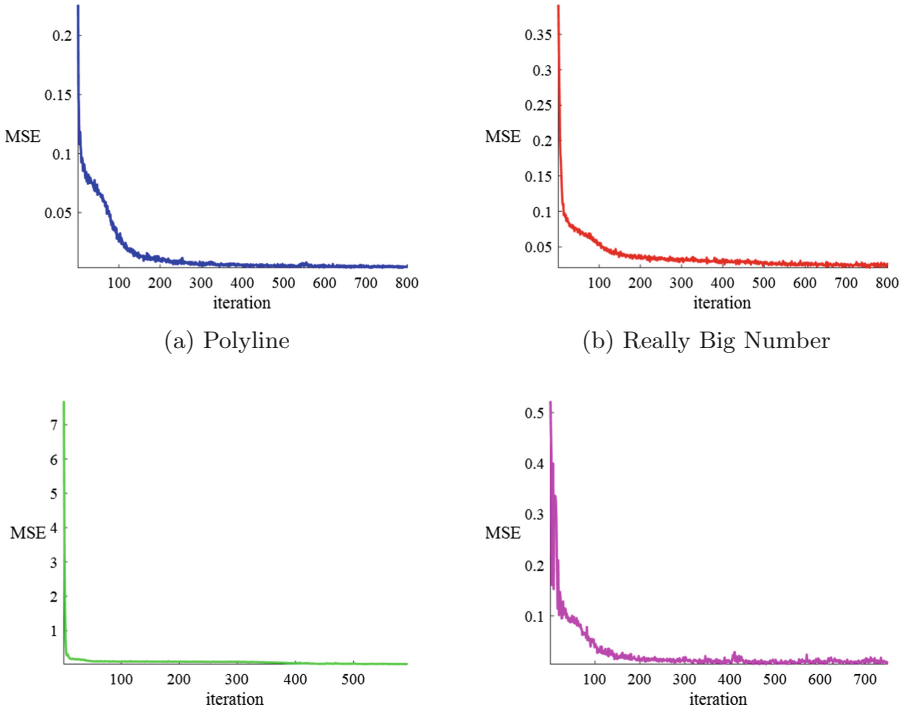
$$MSE = \frac{\sum_{i=0}^{n} \left(y^i - {y'}^i\right)^2}{n} \tag{15}$$

Where n is the total number of data, and y and y' are expected and actual results respectively.

The following figures demonstrate the error value of each individual case study Fig. 4. Looking at the figures in more detail, in each case study by increasing the training iteration, the *MSE* value decreases considerably. It is also noticeable that the error value after a specified iteration converges to the certain value, and after that, by increasing the iteration the *MSE* value fluctuate around that value.

4.3 Assessment of Fault Detection

The aim of the Oracle is finding the existing errors in the under-test software. Meaning that for each test case certify that whether the output of the software is similar to the expected result (Oracle output) or not. To evaluate the function of



(a) Polyline (b) Really Big Number

(c) Magic Number (d) Karen & Neighborhood

Fig. 4. Error per number of iteration

the Oracle, the software source codes were manipulated manually, and for each source code, one line of each code perturbed as follow:

As it is shown in the Table 2, by changing the source code of each program, expected results changes consequently. For instance, in the case of the Polyline program, 617 case of expected results out of 5K test data have been under the influence of our perturbation.

The main task of Oracle is distinguishing between changed and unchanged results. Due to this, the following parameters define as follow:

- True Positive (TP) is the number of test cases which their results have changed, and Oracle could find this change correctly.
- True Negative (TN) is equal to the number of test cases that did not change, and Oracle could figuring out this remain.
- False Positive (FP) define as the number of test cases that did not change, but Oracle knew them as changed cases.
- False Negative (FN) is the test cases that have changed, but Oracle defines them as unchanged results.

Regarding the fact that the value of the actual and expected results never be fully equal, By defining a threshold value, acceptability or unacceptability of

Table 2. Samples of the mutants

#	Original code	Mutated code	Error type	Affected output
1	$if(x_1 == x_2)$	$if(x_1 == x_3)$	Variable change	617
2	$if(temp \leq n)$	$if(temp < n)$	Operator change	811
3	++Digit	Digit++	Operator change	326
4	$if(L\_FB \geq L\_Cnt)$	$if(L\_FB > L\_Cnt)$	Operator change	64

the test case results have evaluated. The threshold in this study defined in the Eq. 16 as follows:

$$|y - y'| < threshold \tag{16}$$

The following table illustrates the values of TP, TN, FP, FN and threshold of the understudy software (Tables 3 and 4):

Table 3. The proposed approach evaluation results

#	Threshold	# TP	# TN	# FP	# FN
1	0.5	583	4263	71	83
2	200	781	4101	83	37
2	20	547	4293	104	56
3	0.5	311	4628	33	28
4	200	51	4929	13	7

Table 4. Effectiveness of purposed method

#	Threshold	P	TPR	FPR	TNR	FNR	ACC	FM
1	0.5	0.89	0.87	0.01	0.98	0.12	0.96	0.88
2	200	0.90	0.95	0.01	0.98	0.04	0.97	0.92
2	20	0.84	0.90	0.02	0.97	0.09	0.97	0.87
3	0.5	0.90	0.91	0.00	0.99	0.08	0.98	0.91
4	200	0.79	0.88	0.00	0.99	0.12	0.99	0.83

For evaluation of the Oracle's results and parameters, the favourable data science formulas have used [14,15], and define as follow:

- Precision: $P = \frac{TP}{TP+FP}$
- True Positive Ratio (Sensitivity or Recall): $TPR = \frac{TP}{TP+FN}$
- False Positive Ratio: $FPR = \frac{FP}{FP+TN}$
- True Negative Ratio (Specificity): $TNR = \frac{TN}{TN+FP}$

– False Positive Ratio: $FNR = \frac{FN}{TP+FN}$
– Accuracy: $ACC = \frac{TP+TN}{TP+TN+FP+FN}$
– F-Measure: $FM = \frac{2\times P\times TPR}{P+TPR}$

5 Conclusion

In this paper, a new approach to designing software test Oracles by the utility of deep learning and fuzzy inference system has presented. To assess the performance of this architecture in action, a software test Oracle designed and tested with four different applications. The performance of this Oracle by importing the error data evaluated. For this purpose, firstly the Oracle trained by correct data, and then by manipulation in source codes of the applications, performance of the Oracle in finding the errors has been assessed. It is found that the Oracle has a high efficiency in detecting the error and also correct data.

The aim of all the proposed test Oracles is detecting the error with the best accuracy and the minimum human contribution in the process for all types of the software. Although the proposed test Oracle has acceptable accuracy in dealing with sample test software, this Oracle has a noticeable weakness. It is important to note that this Oracle can only be used for software that their output is numeric, and it cannot tackle with the string or graphical outputs.

References

1. Ammann, P., Offutt, J.: Introduction to Software Testing. Cambridge University Press, Cambridge (2016)
2. Chen, T.Y., Leung, H., Mak, I.K.: Adaptive random testing. In: Maher, M.J. (ed.) ASIAN 2004. LNCS, vol. 3321, pp. 320–329. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30502-6_23
3. Desikan, S., Ramesh, G.: Software Testing: Principles and Practice. Pearson Education India, New Delhi (2006)
4. Erhan, D., Szegedy, C., Toshev, A., Anguelov, D.: Scalable object detection using deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2147–2154 (2014)
5. Fraser, G., Rojas, J.M.: Software testing. Handbook of Software Engineering, pp. 123–192. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-00262-6_4
6. Gholami, F., Attar, N., Haghighi, H., Asl, M.V., Valueian, M., Mohamadyari, S.: A classifier-based test oracle for embedded software. In: 2018 Real-Time and Embedded Systems and Technologies (RTEST), pp. 104–111. IEEE (2018)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
8. Huang, P.S., He, X., Gao, J., Deng, L., Acero, A., Heck, L.: Learning deep structured semantic models for web search using clickthrough data. In: Proceedings of the 22nd ACM International Conference on Conference on Information & Knowledge Management, pp. 2333–2338. ACM (2013)
9. Jang, J.S.R., et al.: Fuzzy modeling using generalized neural networks and kalman filter algorithm. In: AAAI, vol. 91, pp. 762–767 (1991)

10. Khalilian, A., Mirzaeiyan, A., Vahidi-Asl, M., Haghighi, H.: Experiments with automatic software piracy detection utilising machine-learning classifiers for micro-signatures. J. Exp. Theor. Artif. Intell. **31**(2), 267–289 (2019)
11. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436 (2015)
12. Lee, C.C.: Fuzzy logic in control systems: fuzzy logic controller. I. IEEE Trans. Syst. Man Cybern. **20**(2), 404–418 (1990)
13. Mamdani, E.H., Assilian, S.: An experiment in linguistic synthesis with a fuzzy logic controller. Int. J. Man Mach. Stud. **7**(1), 1–13 (1975)
14. Perruchet, P., Peereman, R.: The exploitation of distributional information in syllable processing. J. Neurolinguistics **17**(2–3), 97–119 (2004)
15. Powers, D.M.: Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation (2011)
16. Ran, L., Dyreson, C., Andrews, A., Bryce, R., Mallery, C.: Building test cases and oracles to automate the testing of web database applications. Inf. Softw. Technol. **51**(2), 460–477 (2009)
17. Schnitman, L., Yoneyama, T.: Takagi-sugeno-kang fuzzy structures in dynamic system modeling. In: Proceedings of the IASTED International Conference on Control and Application (CA 2001), pp. 160–165 (2001)
18. Shahamiri, S.R., Kadir, W.M.N.W., Ibrahim, S.: An automated oracle approach to test decision-making structures. In: 2010 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), vol. 5, pp. 30–34. IEEE (2010)
19. Shahamiri, S.R., Kadir, W.M.N.W., Ibrahim, S., Hashim, S.Z.M.: An automated framework for software test oracle. Inf. Softw. Technol. **53**(7), 774–788 (2011)
20. Shahamiri, S.R., Kadir, W.M.N.W., Mohd-Hashim, S.Z.: A comparative study on automated software test oracle methods. In: Fourth International Conference on Software Engineering Advances, ICSEA 2009, pp. 140–145. IEEE (2009)
21. Sugeno, M., Kang, G.: Structure identification of fuzzy model. Fuzzy Sets Syst. **28**(1), 15–33 (1988)
22. Takagi, T., Sugeno, M.: Derivation of fuzzy control rules from human operator's control actions. IFAC Proc. Vol. **16**(13), 55–60 (1983)
23. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. Syst. Man Cybern. **1**, 116–132 (1985)
24. Watson, A.H., Wallace, D.R., McCabe, T.J.: Structured testing: a testing methodology using the cyclomatic complexity metric, vol. 500. US Department of Commerce, Technology Administration National Institute of Standards and Technology (1996)
25. Whittaker, J.A.: What is software testing? and why is it so hard? IEEE Softw. **17**(1), 70–79 (2000)
26. Zadeh, L.A.: Outline of a new approach to the analysis of complex systems and decision processes. IEEE Trans. Syst. Man Cybern. **1**, 28–44 (1973)
27. Zadeh, L.A., et al.: Fuzzy sets. Inf. Control **8**(3), 338–353 (1965)