SOFTWARE TESTING CHALLENGES IN MACHINE LEARNING APPLICATION

Muhamad Adham Hambali
adhamhambali@kuis.edu.my

Ahmad Nazeer Zainal Arifin
ahmadnazeer@kuis.edu.my

Kolej Universiti Islam Antarabangsa Selangor (KUIS)

ABSTRACT

Machine learning applications have gained excellence results in many parts and produced an effective solution to deal with many issues of sub-field products such as image recognition, automatic driven, voice processing etc. As these machine learning applications are affected by multiple critical interfaces and areas, their reliability and robustness become more crucial. Software testing is the best way to ensure the quality of applications. It is understood that to test machine learning applications is very complicated. This paper discusses fundamentally several major testing challenges on machine learning applications and the solutions recommended from the experts.

Kata Kunci : *Machine Learning Application; Software Testing. Testing Challenges*

INTRODUCTION

Machine learning is an algorithm that learns from past mistakes in order to provide an improved solution. According to John S. Tailor et al, (2004), machine learning is a type of Artificial Intelligence technique that makes decisions or predictions from data. Even though machine learning is a branch of data science, it has gained popularity and achieved results such as self-driving car. Self-driving car cannot be done without machine learning. The algorithm that control the functionality of the car need to understand and adapt the specific situation. That type of system or application like self-driving car is very difficult to develop. One of the headache parts when developing machine learning application is on how to test it properly and smoothly. But until now, the challenges to test it is still being discussed and investigated.

Software testing is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test (Kaner Cem, 2006). Machine learning testing, however, is hard and complicated. It's totally different with traditional software testing. Traditional software testing detects bugs in the code while machine learning testing detects bugs in the data, the learning program and the framework (Jie M. Zhang et al, 2019). According to Carver et al (2017), the difficulties when validating and verifying scientific application like machine learning are:

1. Results are often unknown when exploring novel science or engineering areas and algorithms.
2. Popular software engineering tools often do not work on the architectures used computational science and engineering.

The bugs in machine learning as stated by Jie M. Zhang et al (2019), may exist not only in the learning program but also in the data or the algorithm. Therefore, not just software testers can do the testing, data scientist or algorithm designers could also the role of testers. This paper discussed fundamentally about the selected testing challenges in ML Application and some of the existing recommendations from the current research papers.

MACHINE LEARNING TESTING

Machine Learning testing can be defined as any activity designed to reveal machine learning bugs. The definition of machine bugs and machine learning testing indicate three aspects of machine learning: the required conditions, the machine learning items and the testing activities (Jie M. Zhang et al, 2019). The process of traditional software testing is normally straight forward. Bugs usually can be detected in the code. In machine learning testing, it is not only the code that may contain bugs but also the data. The hardest part in testing machine learning application based on the papers and articles published is test oracle. Test coverage, white box testing, and test adequacy are among testing challenges that also been highlighted in current academic papers.

Song Huang et al, (2018) mentioned in their paper, the rule and logic of traditional application are expressed in terms if control flow, which are encoded by programmers. When an input is received, a certain operation is used to transform a statement to another. In traditional software testing application, it is possible to extract an oracle. When conducting ML application testing, the methods implemented for traditional application may become invalid. They also stated in their paper, regarding the test coverage. Several metrics such as statement coverage, branch coverage, condition coverage and modified condition or decision coverage are effective in testing ML applications in code level. Unlike traditional application testing, in ML application testing, a few test inputs can achieve high statement coverage. However, Song Huang et al (2018) mentioned, only a low percentage of the ML application is accessed.

In D Marijan et al (2019), they stated that the technique of white box testing for ML application requires high test effort. It can be costly due to time-consuming and labour-intensive process.

SELECTED CHALLENGES AND RECOMMENDATIONS OF TESTING ML APPLICATION

This paper only focuses on the discussions of important selected challenges of testing ML application. This paper also fundamentally explains about the existing solutions recommended from the experts.

Test Oracle Problem

The term test oracle was introduced by W. E. Howden (1978). In software testing, an oracle is a mechanism to determine whether a test case passed or failed. It also can refer to a technique to tell whether a program is working properly. However, in machine learning testing, due to many machine learning algorithms are probabilistic programs, it is hard to determine the correctness of the output (D. Marijan et al, 2019). Pomin Wu, (2017) stated in his article, the issue in machine learning is that there is no oracle without human involvement.

Machine learning based-software is considered non-testable due to lack of test oracle. In order to test non-testable software, researchers have used pseudo-oracles (M D. Davis et al, 1981). The idea of pseudo-oracles is creating multiple independent implementations of same algorithm and comparing the outputs of these implementations according to same input. If the outputs are having discrepancies, there may be a defect in one or more implementations.

Another technique to create test oracles is metamorphic testing. The initial idea of metamorphic testing came from T.Y. Chen (1998). Basically, metamorphic testing is a technique to generate follow-up test cases based on existing test cases that have not disclosed any failure. Metamorphic testing should be implemented in conjunction with other test case selection strategies that create the initial set of test cases (Zhou et al 2004). Metamorphic testing is an effective technique to produce test oracles. However, according to S. Huang et al (2018), the generality of metamorphic relationships is low.

White Box Testing

White box testing is a technique to test internal structure of the system. Meaning that tester needs to test the quality of the code. White box testing is the best suited for algorithm, therefore, it is the best technique to be used to test machine learning application. However, it is not easy to test machine learning application thus it will take time and can be costly.

Researchers have proposed one black box testing technique to replace white box testing in order to test machine learning application. The technique is called adversarial testing which uses perturbations inputs to generate adversarial examples for testing the robustness of machine learning models (D. Marijan et al, 2019). Although, most of research papers have focused only for adversarial testing technique on image classification, a novel research area is arising on producing adversarial examples for non-image data (N. Carlini et al, 2018).

Test Coverage

Test coverage for traditional software testing is to measure the degree to which the source code of a program is assessed with the supplies test inputs (S. Nakajima et al 2016). Several metrics have been proposed to measure test coverage such as statement coverage, condition coverage, branch coverage etc (G. J, Mayers et al, 1979). In machine learning application testing, these metrics are effective in code level. Different form traditional application, in machine learning application testing, even a few test inputs can reach high statement coverage. However, only a low percentage of the application is evaluated.

Pei et al (2017) has proposed neuron coverage for the deep neuron networks as the new metric. It can be computed as equation below:

$$neuron\ coverage = \frac{number\ of\ activated\ neurons}{number\ of\ total\ neurons}$$

Equation by S. Nakajima et al 2016

An activated neuron is the neuron whose output is larger than the hyperparameter: threshold. The higher neuron coverage is, the more weights, learned from training data, can be evaluated. Generalizing other ML applications, the new metric should indicate the degree of how much of the knowledge learned from the data assessed (S. Huang et al 2018).

CONCLUSION

Testing of machine learning application is difficult and challenging because it's totally different with traditional software testing in terms of testing technique, bug detection and also the role of testing. Test oracle remains the main problem of ML, but the results of the proposed recommendations are promising.

REFERENCES

D. Marijan, A. Gotlieb and M. Kumar Ahuja, "Challenges of Testing Machine Learning Based Systems," 2019 IEEE International Conference on Artificial Intelligence Testing (AITest), Newark, CA, USA, 2019, pp. 101-102, doi: 10.1109/AITest.2019.00010.

Elaine J. Weyuker, On Testing Non-Testable Programs, The Computer Journal, Volume 25, Issue 4, November 1982, Pages 465–470, https://doi.org/10.1093/comjnl/25.4.465

E. T. Barr, M. Harman, P. McMinn, M. Shahbaz and S. Yoo, "The Oracle Problem in Software Testing: A Survey," in IEEE Transactions on Software Engineering, vol. 41, no. 5, pp. 507-525, 1 May 2015, doi: 10.1109/TSE.2014.2372785.

G. J. Myers and C. Sandler, The Art of Software Testing: Wiley, 1979

John Shawe-Taylor, Nello Cristianini, et al. Kernel methods for pattern analysis. Cambridge university press, 2004.

J. M. Zhang, M. Harman, L. Ma and Y. Liu, "Machine Learning Testing: Survey, Landscapes and Horizons," in IEEE Transactions on Software Engineering, doi: 10.1109/TSE.2019.2962027.

J. Carver, N. P. C. Hong, and G. K. Thiruvathukal, Eds., Software engineering for science. Boca Raton: Taylor & Francis, CRC Press, 2017.

M D. Davis and E. J. Weyuker, Pseudo-oracles for non-testable programs, In Proc. of the ACM '81 Conference, pp. 254-257t, 1981.

N. Carlini and D. Wagner, "Audio Adversarial Examples: Targeted Attacks on Speech-to-Text," 2018 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, 2018, pp. 1-7, doi: 10.1109/SPW.2018.00009.

Pomin Wu, 2017, https://medium.com/trustableai/testing-ai-with-metamorphic-testing-61d690001f5c, retrieved date July 17 2020.

S. Nakajima and H. N. Bui, "Dataset Coverage for Testing Machine Learning Computer Programs," in 2016 23rd Asia-Pacific Software Engineering Conference, A. Potanin, G. C. Murphy, S. Reeves, and J. Dietrich, Eds., ed New York: Ieee, 2016, pp. 297-304.

T.Y. Chen, S.C. Cheung, and S.M. Yiu, "Metamorphic testing: a new approach for generating next test cases", Technical Report HKUST-CS98-01, Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong (1998).

W. E. Howden, "Theoretical and Empirical Studies of Program Testing," in IEEE Transactions on Software Engineering, vol. SE-4, no. 4, pp. 293-298, July 1978, doi: 10.1109/TSE.1978.231514.

Zhou ZQ, Huang D, Tse T, Yang Z, Huang H, Chen T. Metamorphic testing and its applications. Proceedings of the 8th International Symposium on Future Software Technology (ISFST 2004): Xian, China, 2004; 346–351.