2019 IEEE International Conference on Artificial Intelligence Testing (AITest)

Challenges of Testing Machine Learning Based Systems

Dusica Marijan
Simula Research Laboratory
Norway
dusica@simula.no

Arnaud Gotlieb
Simula Research Laboratory
Norway
arnaud@simula.no

Mohit Kumar Ahuja
Simula Research Laboratory
Norway
mohit@simula.no

***Abstract*—Machine learning is being increasingly deployed in many industrial and societal applications. Given its widespread use, it is of utmost importance to ensure the quality of software systems supported by machine learning. Although researchers have applied some concepts from traditional software testing to testing of machine learning based systems, the latter introduce a range of challenges not typical for traditional software systems, thus making traditional software testing techniques ineffective. In this paper, we discuss the challenges intrinsic to testing of machine learning based systems. We highlight the promising role of machine learning based testing to alleviate some of these challenges. We also discuss directions for future research in this domain. This paper focuses on the testing aspects of machine learning based systems from the quality assurance perspective, rather than model performance perspective.**

***Index Terms*—software testing, machine learning, machine learning based testing, testing of machine learning based systems**

I. Introduction

Machine learning (ML) has gained popularity in many application areas, including safety critical domains. However its vulnerability has recently started surfacing, sometimes leading to catastrophic failures [1]. This entails that comprehensive testing of ML-based systems needs to be performed, to ensure the correctness of these systems, and to safeguard their users and environment against the malfunctions.

Testing of ML-based systems is characterized by a number of challenges compared to testing of traditional software systems. By *traditional systems* we mean systems not using ML, and by *ML-based systems* we mean intelligent systems supported by ML (e.g collaborative robots, autonomous vehicles). The most notable challenge in testing ML-based systems stems from non-determinism intrinsic to ML. While traditional systems are typically pre-programmed and execute a set of rules, ML-based systems reason in a probabilistic manner and exhibit non-deterministic behavior. For example, for constant test inputs and preconditions, a ML-based system can produce different output in each run. To address some of these and related challenges, researchers have used testing techniques from traditional software development, such as coverage metrics and combinatorial testing techniques. However, it has been observed that traditional testing approaches in general fail to adequately address fundamental challenges of testing ML-based systems [2], and that these approaches require adaptation to the new context of ML.

This paper provides a set of selected challenges in testing ML-based systems, with the goal of forming a better understanding of the research challenges that require attention, contributing to new ideas, and encouraging new research directions in this field. We provide a brief overview of some of the existing attempts to address some of these challenges, as discuss their limitations. We highlight the role of ML-based testing in addressing these challenges.

Twofold interpretation of the term *testing*. The concept of testing ML systems is often discussed in two scientific communities, namely, the ML community (MLC) and the software testing community (STC). However, as two communities interpret the term *testing* differently, it is worth noting the distinction. In MLC, testing a ML model is performed to estimate its prediction accuracy, and improve its prediction performance. In this context, testing happens during a model development, by comparing the output values predicted by the model with actual values. In STC, testing a ML-based system denotes checking the system behavior for any given inputs, including those for which the expected outputs are unknown. Furthermore, in case of integration or system level testing of a ML-based system, a ML component is tested in interaction with other system components, for a range of quality attributes, including correctness, robustness, or reliability. This paper focuses on the latter of the two interpretations.

II. Selected Challenges of Testing ML Systems

Due to space constraints, this paper focuses on the selected most prominent challenges of testing ML-based systems. Along with the challenges, we review existing approaches, discuss their limitations and highlight potential future research.

A. Absence of Test Oracles

In testing traditional systems, a common assumption is that there exists a mechanism for assessing the correctness of the output produced by the software under test [3], known as test oracle. However, because ML-based systems reason probabilistically, their output is learned and predicted by a ML model, rather that specified prior to testing. This means that ML-based systems do not have defined expected values against which actual values can be compared in testing. Thus the correctness of the output in testing a ML-based system cannot be easily determined.

978-1-7281-0492-8/19/$31.00 ©2019 IEEE
DOI 10.1109/AITest.2019.00010

The challenge of the lack of test oracles has been observed before, with traditional software systems. Such systems were considered non-testable [3]. To test non-testable software systems, researchers have used pseudo-oracles [4], which are a differential testing technique that consists in running multiple software programs satisfying the same specification as the original software under test, then feeding the same inputs to these software programs and observing their outputs. Discrepancies in outputs are considered indicative of errors in the software under test. However, differential testing can be resource-inefficient as it requires multiple runs of the software, and error-prone, as the same errors are possible in multiple implementations of the software under test [5]. Applying a similar concept of differential testing, DeepXplore [7] has been proposed to detect errors in deep neural networks (DNN) by introducing perturbations in input data. Another approach to testing of software without test oracles has been metamorphic testing [6]. In this approach, a transformation function is used to modify the existing test case input, and produce the new output. If the actual output differs from the expected output, it is indicative of errors in the software under test. However, in testing ML-based systems with a large input space, writing transformations is laborious, and there is a great potential for ML to circumvent this difficulty by automating the creation of metamorphic relationships.

B. Large Input Space

ML-based systems are usually deployed in application areas dealing with a large amount of data. This creates a voluminous and diverse test input space. Specifically, the difficulty lies in systematically selecting an adequate set of inputs and their combinations from a large input space, which are able to trigger system faults in testing. Furthermore, large input spaces complicate the test oracle problem discussed previously, because specifying expected outputs for all valid system inputs is intractable.

To deal with this challenge, researchers have applied coverage metrics to testing of neural networks, inspired by the traditional code coverage metrics. For example, the metric called neuron coverage proposed in DeepXplore [7] measures the amount of neurons activated by a set of inputs. This approach has been further extended in DeepGauge [8], which aims to test DNN by combining the coverage of key function regions as well as corner case regions of DNN, represented by neuron boundary coverage. Other approaches were proposed extending the notion of neuron coverage. These approaches include DeepTest [9] for testing other types of neural networks, and DeepRoad [10], which uses Generative Adversarial Networks to generate test cases for DNN in autonomous cars.

However, this class of techniques based on neuron coverage can easily lead to combinatorial explosion, due to a large test input space. Initial work on the adaptation of combinatorial testing techniques for the systematic sampling of a large space of neuron interactions [11] provides a promising direction for further research effort on taming combinatorial explosion in testing of software systems that use DNN.

C. White Box Testing Requires High Test Effort

Comprehensive testing of ML-based systems using white box techniques discussed previously can be costly, and therefore researchers have proposed black box testing techniques for ML-based systems. One such popular technique is adversarial testing, which uses perturbations in inputs to generate adversarial examples for testing the robustness of ML models. While the mainstream research efforts on adversarial testing have focused on testing image classification problems, a novel research area is emerging on generating adversarial examples for non-image data, for example, text and audio [12]. Specifically, because adversarial examples are often not sufficiently representative of real inputs, there is a potential for ML-based testing techniques to improve the effectiveness of adversarial examples and thus increase the effectiveness of detecting errors in real scenarios for ML-based system testing.

III. Discussion and Conclusion

Testing of ML-based systems is challenged by a number of difficulties, including the lack of test oracles, voluminous test input spaces, and high test effort of comprehensive testing, to name a few. A promising approach to addressing these challenges is the adaptation of traditional testing techniques with the support of ML. Specifically, ML for the automated generation of metamorphic relationships, generation of realistic adversarial examples (also for non-image data), and efficient sampling of input data with combinatorial interaction coverage. ML-based testing has the potential to increase cost-effectiveness and scalability of testing, especially for ML-based systems operating in highly non-deterministic environments and handling large amounts of data.

References

[1] Tesla failure, 2016, Retrieved on 20 December 2018 from www.theguardian.com/technology/2016/jul/01/tesla-driver-killed-autopilot-self-driving-car-harry-potter

[2] C. Hutchison, M. Zizyte, P. E. Lanigan, et al, Robustness testing of autonomy software, SEIP at Int. Conf. on Soft. Eng, 2018, pp. 276-285.

[3] E. J. Weyuker, On testing non-testable programs, Computer Journal, 25(4), pp. 465-470, 1982.

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

[5] J. Knight and N. Leveson, An experimental evaluation of the assumption of independence in multi-version programming, IEEE Transactions on Software Engineering, 12(1), pp. 96-109, 1986.

[6] F. Chan, T. Chen, et. al, Application of metamorphic testing in numerical analysis, IASTED Conf. in Software Eng., 1998, pp. 191-197.

[7] K. Pei, Y. Cao, J. Yang, S. Jana, Deepxplore: Automated white-box testing of deep learning systems, 26th Symposium on Operating Systems Principles, ACM, 2017, pp. 1-18.

[8] L. Ma, F. Juefei-Xu, F. Zhang, J. Sun, M. Xue, B. Li, C. Chen, et al., Deepgauge: multi-granularity testing criteria for deep learning systems, ACM/IEEE Int. Conf. on Automated Software Eng., 2018, pp. 120-131.

[9] Y. Tian, K. Pei, S. Jana, B. Ray, Deeptest: Automated testing of deep-neural-network-driven autonomous cars, International Conference on Software Engineering, ACM, 2018, pp. 303-314.

[10] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, Deep-road: Gan-based metamorphic autonomous driving system testing, Int. Conf. on Automated Soft. Eng., 2018, pp. 132-142.

[11] L. Ma, F. Zhang, M. Xue, B. Li, Y. Liu, J. Zhao, Y. Wang, Combinatorial testing for deep learning systems, arXiv preprint arXiv:1806.07723.

[12] N. Carlini, D. Wagner, Audio adversarial examples: Targeted attacks on speech-to-text, IEEE Security and Privacy Workshop, 2018.