CHALMERS | UNIVERSITY OF GOTHENBURG

Department of Computer Science and Engineering, Agneta Nilsson
Software Engineering Master Thesis Proposal, 30hec

Benchmarking Deep Learning Testing Techniques: A Methodology and Its Application

Name 1: Himanshu Chuphal

Completed courses relevant for thesis work:
Software Evolution Project
Empirical Software Engineering
Model-Based Testing
Software Quality

Name 2: Kristiyan Dimitrov

Completed courses relevant for thesis work:
Software Evolution Project
Empirical Software Engineering
Software Quality

1 Introduction

Over the past few years, Deep Learning (DL) systems have made rapid progress, achieving tremendous performance for a diverse set of tasks, which have led to widespread adoption and deployment of Deep Learning in security and safety-critical domain systems [1][7]. Some of the most popular examples include self-driving cars, malware detection, and aircraft collision avoidance systems [8]. Due to their nature, safety-critical systems undergo rigorous testing to assure correct and expected software behavior. Therefore testing of DL systems becomes crucial, which has traditionally only relied on manual labeling/checking of data [7].

There are multiple DL testing techniques to validate different parts of a DL system's logic and discover the different types of erroneous behaviors. Some of the more popular DL testing techniques [1] [2] [8] [7] [9], which validate deep neural networks using different approaches for fault detection. While ensuring predictability and correctness of the DL systems to a certain extent, there is no standard method to evaluate which testing technique is better in terms of certain testing properties, i.e. test coverage, correctness, robustness, etc.

There is a guide for the selection of appropriate hardware platform and DL software tools [4] but no guide available as such to select appropriate DL testing techniques. Additionally, there is an increasing trend in the research work done within the field of DL testing. Therefore, there is a need for a method to benchmark DL testing tools

with in-depth analysis. The method will serve as a guide to practitioners/testers and the DL systems community.

2 Statement of the problem

DL systems are rapidly being adopted in safety and security-critical domains, urgently calling for ways to test their correctness and robustness. Consequently, there is an increase in the cumulative number of publications on the topic of testing machine learning systems between 2018 and June 2019, 85% of papers have appeared since 2016, testifying to the emergence of new software testing domain of interest: machine learning testing [5]. Recently, a number of good DL testing tools have been proposed. However, there is no guide available as such to select an appropriate DL testing techniques based on *testing properties* (e.g., correctness, robustness, and fairness), *testing components* e.g., the data, learning program, and framework), *testing workflow* (e.g., test generation and test evaluation), and *application scenarios* (e.g., autonomous driving, machine translation) [5]. Since, there is no standard method to evaluate DL testing tools, the selection of an appropriate DL testing tool or technique for a relevant DL use case remains a challenge for practitioners.

3 Purpose of the study

The purpose of the study is to design a method for benchmarking DL testing tools. The benchmark method shall be constructed following the guidelines proposed by Sim et al. [10] for an academic purpose. The selected testing techniques will be evaluated using a sufficiently complex task sample as to bring out their effectiveness. The method would provide academia with a standard which can be used to verify the effectiveness of both existing and newly aspiring DL testing techniques. Afterward, we hope to receive feedback from at least one industry source on the usability of the method within an industrial setting and use that feedback to extend the method. In addition, the results of the study can be used in itself by industrial experts to support the selection of an appropriate DL testing technique.

4 Review of the literature

There is an increasing trend in the research work done within the field of DL testing methods and there is no guide available as such to select appropriate DL testing techniques. Some of the most popular techniques to validate DL systems include *DeepXplore* [1], *DeepTest* [2], *DLFuzz* [8], *Guiding deep learning system testing using surprise adequacy* [7], *Testing Deep Neural Networks* [9], *etc.*.

Most testing techniques measure training input coverage of the DL system. The two assumptions that most testing techniques follow are [7]:

- if two inputs to a DL system are similar to a human sense then the output should also be similar and
- the more diverse a set of inputs is the better the DL system will perform.

For example, *DeepTest* [2] checks whether an autonomous driving system behaves in the same way when the input image is transformed as if the same scene is under a different weather condition. Another good example is of *DeepXplore* [1], which presented the Neuron Coverage (the ratio of neurons whose activation values were above a predefined threshold) as the measure of diversity of neuron behaviour, and subsequently showed that inputs violating the first assumption will also increase the neuron coverage.

Existing testing techniques mainly focus on identifying test inputs that improve a given test coverage metric for test generation. The exploits of DL and it's movement within the safety-critical sector have become apparent within the testing community: both adequacy testing [7] and DLFuzzing [8] recognize that focus on testing needs to be done within the area. The empirical study and adequacy testing both recognize the progress made towards neuron coverage but point out other issues which cannot be solved easily by it. The empirical study proposes that there is a lack of testing frameworks in regards to differential testing of model migration. Adequacy testing [7] focuses on corner case input and their impacts and proposes a solution to it. All papers point out different properties which are missing and those that are covered. Papers focused on Technique development solve continuously segments of the "missing" testing properties but there is no formal benchmark and classification of those techniques, all are spread out within research articles.

DLFuzzing [8] supports the statement of adequacy that most techniques take constraints in the form of assumptions about the DL system. Whitebox testing techniques are accurate but very resource-consuming (we can use this to justify the need for benchmarking). An example is DeepXplore [1] that does not require manual labeling by using a similar DL system as a cross-referencing oracle. This is clearly resource consuming because of the difficulty of finding a similar DL system. Additionally cross-referencing suffers from scalability. Blackbox models, on the other hand, are time-consuming and rely heavily on manual labeling supplies. DLFuzzing [8] essentially does what DeepXplore [1] does but faster and better in terms of neuron coverage and adversarial inputs according to the paper. Adequacy testing used DeepXplore, it is popular but maybe not the best, can use this to support our claim for need of benchmarking. Both Adequacy and DLFuzzing point towards a way of testing that's called "adversarial" testing.

5 Research question and/or Hypotheses

Our benchmarking evaluation is designed to answer the following 3 research questions (**RQ1-3**):

RQ1: What existing state-of-the-art and real-world applicable DL testing tools are available? We investigate the correctness, fairness, efficiency, and robustness properties of the existing DL testing tools and decide which testing tools are most suited for a relevant case.

RQ2: *To what extent do different testing techniques and workflows of the DL testing tools perform better towards DL application scenarios?*
Our goal is to design a methodology that uses experimental tests with a relevant DL system using DL testing tools for diverse datasets and benchmark the results. Check the possibility to get feedback from industry experts and check the applicable improvement in the method?
***Hypothesis 1*:** There are significant qualitative differences in the selected DL testing tools in terms of testing properties.
***Hypothesis 2*:** There is a significant difference in test input generation of selected DL testing tools for a given type of dataset.

RQ3: *How can the proposed benchmarking methodology of DL testing tools help practitioners and possibly industry experts as a reference for selecting an appropriate testing technique?*
We aim is to assess the benchmarking methodology using different DL testing tool for an industry-grade dataset. The benchmarking methodology will be presented to two industry contacts to receive feedback and assess the feasibility of its application in real-world DL scenarios.

6 The Design – Methods and Procedures

The intended research involves study of literature review and design research of a benchmarking method along with an experiment for the evaluation either in a commodity laptop, conducted **SNIC (**Swedish National Infrastructure for Computing **)** setup or Chalmers IT setup. For the overall design of the study, we focus on the proposed five requirements for creating a successful benchmark within software engineering: clarity, relevance, solvability, portability, and scalability [10]. As such the study will have characteristics of both formal experiments and case studies.

Relevance: The relevance requirement in the case of the DL testing tool benchmark is related to the task sample as defined in *sub-section 1*. The datasets and DNN models need to be representative of actual data that the system is expected to handle.

Solvability: Similar to the relevance requirement, the solvability requirement is concerned with the task sample. The data sets and DNN need to comprise a task that is not too difficult for the DL testing tools to handle, as well as being not too simple for the tools to show their potential as best as possible.

Portability: The portability aspect in the case of the DL benchmark is concerned with the systems and architecture on which the DL system will be run, as well as the system itself. Due to resource restrictions, the benchmark will be constructed to work using the available hardware and software.

Scalability: This requirement involves the benchmark being able to work with as wide a variety of DL testing tools as possible. The initial benchmark will be meant to work with the currently available DL testing tools and possibly extended with feedback from an industry expert to support industrial purposes.

Clarity: The clarity requirement involves the understandability and conciseness of the benchmark. This requirement will be tested if an optional industry contact is found. Through the feedback, we'll discover if the benchmark's clarity requirement is fulfilled.

1. First, a selection phase shall be conducted. A study of related research and gray literature will be done on the most popular DL testing tools to investigate the most appropriate DL testing tools (**RQ1**). After the relevant tools based on the criteria defined in RQ1 are identified a random sample of 3, at most 4, testing tools will be selected to be used in the comparison. Next, an appropriate hardware platform will have to be selected on which the DL software tools will be run because DL systems require a considerable amount of computational power. Additionally, DL systems require an open-source DNN model on which tests can be run, as such, we have decided to select a minimum of three supervised DNN models for two reasons. First, to set a common ground for all DL tests and second, to center the focus on the benchmarking test results rather than on the models and execution of the testing tools. Finally, a minimum of three open-source datasets will be selected as input for the tests, this is very important for the relevance of the benchmark. Having defined three DNN models and three datasets will allow the construction of a *3x3* task sample.

2. Once the selection phase is over and the relevant task sample has been decided on, representative performance measures will need to be established. For this purpose, further research into the relevant attributes to be measured will have to be conducted. It cannot be said in advance what measurements can be used due to a lack of obvious performance measures prior to the benchmarking effort within a

field as complex as DL testing. This problem has been identified as common within software engineering [10].

3. After the task sample and performance measures have been established, the tests can be conducted. The tests will be executed on a commodity laptop or in SNIC or Chalmers IT DL setup. Each appropriate testing tool will be executed against the predefined task sample in the selection phase. Afterward, the results of the benchmark will be documented by filling out the performance measurement metrics. Then the results of the testing tools will be used for the comparison analysis.

4. Next is the optional feedback gathering and refinement of the method. One feedback session with industry experts will be conducted. In that session, the method shall be presented to the experts and asked for feedback on how the method could be improved to facilitate industry needs in terms of DL testing tool selection. This will improve the scalability of the benchmark to not only include research prototypes but also industrial use. The step is optional because industry collaboration cannot be ensured at this stage. Once the feedback has been gathered, the feasibility of adopting the feedback to the method shall be assessed and subsequently implemented if it can indeed be adopted within the scope of the study. Afterward, *step 3* will be repeated and the new results documented.

Time schedule.
The available time comprises about 20 weeks total, from the mid of December 2019 to late May 2020.

- Extensive research into the newest relevant papers. *2 weeks.*
- Selection of DL testing tools on a relevant case. *3 weeks.*
- Identification of performance measures. 3 *weeks.*
- Execution of testing tools. *6 weeks.*
- Documentation of benchmarking results. *3 weeks.*
- Possible Industry feedback and incorporation. *3 weeks.*

7 Limitations and Delimitations

The development of the design method for the selection of a DL testing tool and benchmarking the performance measure is a shared effort in the research collaboration. The thesis is intended to be limited to the assessment of three DL testing tool on appropriate datasets. Currently, we are in discussion with two industry contacts to apply our designed methodology on a relevant industry-grade DL use-case. Based on the progress of the discussion, **RQ3** could be revised with the supervisor.

Threats to internal validity.
Benchmarks have been used in computer science to compare the performance of computer systems, information retrieval algorithms, databases, and many other technologies. The creation and widespread use of a benchmark within a research area is frequently accompanied by rapid technical progress and community building. The identification of relevant performance measures for benchmark is a crucial phase, which means if the measures established in the design step 2 don't cover relevant testing properties (e.g., correctness, robustness, and fairness), benchmarking results won't be of any significance when it comes to selection of appropriate DL testing tools. An early careful identification of relevant measures at the current time would mitigate the issues of wrong measurements before the control experiment.

All the tests will be conducted on a commonly available hardware platform (laptop, SNIC or Chalmers IT setup). The results would risk the portability, which is local to benchmarking designs and introduce biases. A common hardware (having a fixed number of CPUs and GUPs, subjected to availability of resources) and architecture platform common for all the DL testing techniques would set the common testing ground for benchmarking results.

Threats to external validity.
The benchmarking of the DL testing tool should be both representative and generalizable. Due to a large number of public DL models and diverse datasets, there will be limited control over the selection of DL software testing tools and public datasets. To make it generalizable, at least a minimum of three DNN models and three datasets will be selected as input for the tests.

Performance measures are also a concern in generalizability. Having an insufficient number of performance measures would reduce the applicability of the benchmark. To prevent this, as stated in the internal validity, a significant amount of time is dedicated to research and identification of the current measurements. Additionally, we do not take into account predictions of the direction that measurements of DL testing tools will take due to the expectancy of the research community to iterate and maintain the benchmark in line with the new standards [10].

8 Significance of the study

The result of the benchmarking method will be a comparison of different DL tools and techniques, this on its own will bring awareness to the DL researchers not only of their own work but also of the work and contribution of other researchers within the field. Becoming more aware of each other's work will strengthen the ties of researchers with similar interests within the field [10]. Additionally, if the possibility of industry feedback

bears fruit, the results of the study can be used by industry experts to help in the selection of an appropriate DL testing technique.

9 References

[1] K. Pei, Y. Cao, J. Yang, and S. Jana, *"DeepXplore: Automated whitebox testing of deep learning systems," in Proceedings of the 26th Symposium on Operating Systems Principles. ACM,* 2017, pp. 1–18.

[2] Y. Tian, K. Pei, S. Jana, and B. Ray, *"DeepTest: Automated testing of deep-neural-network-driven autonomous cars," in Proceedings of the 40th International Conference on Software Engineering. ACM, 2018,* pp. 303– 314.

[3] Y. Sun, X. Huang, and D. Kroening, *"Testing deep neural networks,"* arXiv preprint arXiv:1803.04792, 2018.

[4] S. Shi, Q. Wang, P. Xu, and X. Chu, *"Benchmarking state-of-the-art deep learning software tools,"* arXiv preprint arXiv:1608.07249v7 [cs.DC], 2016

[5] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, *"Machine Learning Testing: Survey, Landscapes and Horizons,"* arXiv e-prints, Jun 2019.

[6] F. Ferreira, L. Lourdes Silva, and M. Tulio Valente, "*Software Engineering Meets Deep Learning:A Literature Review,"* arXiv:1909.11436v1 [cs.SE] 25 Sep 2019

[7] J. Kim, R. Feldt, S. Yoo,"*Guiding deep learning system testing using surprise adequacy"* arXiv:1808.08444 (2018)

[8] J. Guo, Y. Jiang, Y. Zhao, Q. Chen, and J. Sun. 2018. *"DLFuzz: differential fuzzing testing of deep learning systems".* In ESEC/SIGSOFT FSE.

[9] Y. Sun, X. Huang, and D. Kroening, *"Testing Deep Neural Networks,"* ArXiv e-prints, Mar. 2018.

[10] Sim S, Easterbrook S, Holt R (2003) "*Using benchmarks to advance research: a challenge to software engineering. In: Proceedings of the 25th International Conference on Software Engineering."* Portland, Oregon.