Empirical Software Engineering (2023) 28:74
https://doi.org/10.1007/s10664-023-10326-7

Software testing in the machine learning era

Special issue of the empirical Software Engineering (EMSE) journal

Andrea Stocco[1,2] · Onn Shehory[3] · Gunel Jahangirova[4] · Vincenzo Riccio[5] · Guy Barash[6] · Eitan Farchi[7] · Diptikalyan Saha[8]

Published online: 4 May 2023

Preface Machine Learning (ML) and Deep Learning (DL) have become ubiquitous in modern software systems, including safety-critical domains such as autonomous cars, medical diagnosis, and aircraft collision avoidance systems. Thus, it is crucial to rigorously test such *learning-based applications* to ensure high reliability and dependability. However, traditional notions of software quality and reliability become obsolete when dealing with learning-based systems, due to their non-deterministic nature and the lack of a transparent comprehension of the models' semantics. The impact of ML and DL extends beyond offer-

✉ Andrea Stocco
stocco@fortiss.org; andrea.stocco@tum.de

Onn Shehory
onn.shehory@biu.ac.il

Gunel Jahangirova
gunel.jahangirova@kcl.ac.uk

Vincenzo Riccio
vincenzo.riccio@uniud.it

Guy Barash
guy.barash@wdc.com

Eitan Farchi
farchi@il.ibm.com

Diptikalyan Saha
diptsaha@in.ibm.com

1 fortiss GmbH, Guerickestraße 25, 80805 Munich, Germany

2 Technical University of Munich, Boltzmannstraße 3, 85748 Garching near Munich, Germany

3 Bar Ilan University, Ramat Gan 5290002, Israel

4 King's College London, Bush House 30 Aldwych, London WC2B 4BG, UK

5 University of Udine, via delle Scienze 206, Udine 33100, Italy

6 Quai.MD, Tel Aviv, Israel

7 IBM Research, Israel, University of Haifa Campus, Mount Carmel, Haifa 3498825, Israel

8 IBM Research, India, Pvt. Ltd. G2, 8th Floor, Manyata Tech Park, Bangalore 560045, India

ing new applications and case studies for testing techniques. In fact, they are revolutionizing the way software is developed and tested. ML and DL are increasingly being utilized to devise novel program analysis and software testing techniques for malware detection, fuzz testing, bug-finding, and type-checking.

1 The special issue

The aim of this special issue is to provide an introduction to the burgeoning field of software testing in the machine learning era. Machine Learning (ML) and Deep Learning (DL) have been widely adopted in modern software systems, including safety-critical domains such as autonomous cars, medical diagnosis, and aircraft collision avoidance systems. It is therefore crucial to rigorously test such *learning-based applications* to ensure high reliability and dependability. While traditional notions of software quality and reliability become inapplicable, the need for a more transparent comprehension of these models' semantics demands further research investigation in the domain of testing ML/DL software systems.

This issue of the Empirical Software Engineering (EMSE) journal is devoted to the intersection of Software Engineering (SE) and ML/DL. Following an open call for papers, the four accepted articles cover various areas within this theme, ranging from novel techniques to ensure the quality of learning-based applications to novel techniques that employ ML or DL to support software engineering tasks. The special issue was preceded by an international workshop on Testing for Deep Learning and Deep Learning for Testing (DeepTest), held virtually on June 1st, 2021.

The articles have undergone rigorous peer review, in accordance with the journal's high standards. In total, six submissions were received and each submission went through a rigorous reviewing process where they received three reviews by different guest editors and were carefully discussed until a consensus was reached. All decisions were based solely on the quality of the submissions, with no specific number of papers targeted for acceptance. Ultimately, four papers were accepted by the guest editors and are briefly discussed below:

In one paper, the authors present DiverGet, a search-based testing framework for quantization assessment. Quantization is one of the most applied Deep Neural Network (DNN) compression strategies, for deploying a DNN model on a resource-constrained system and traditional testing methods are inadequate for quantized DNNs. DiverGet uses metamorphic relations to detect disagreements among DNNs of different arithmetic precision. Particularly notable, DiverGet targets behavioural divergence maximization through the exploration of the vector space. DiverGet outperforms the existing DiffChaser technique when testing test DNNs for hyperspectral remote sensing images, used in critical domains like climate change research and astronomy. In this domain, the searching strategy based on population-based metaheuristics outperforms random search with Particle Swarm Optimization showing higher effectiveness than a Genetic Algorithm.

In another paper, the authors explore the increasing demand for reliable Machine Learning (ML)-based systems in safety-critical fields. To evaluate the reliability of these systems, a benchmark of bugs in ML-based systems, known as a fault load benchmark, is crucial. The study reviewed 1777 ML-related bugs from GitHub and 1296 from Stack Overflow that pertained to ML-based systems utilizing the two most popular ML frameworks, TensorFlow and Keras. The study found that only 3.48% of GitHub bugs and 2.93% of reported bugs in Stack Overflow are reproducible. In response to these findings, the article proposes a fault load benchmark of ML-based systems that consists of 100 bugs extracted from software systems that use TensorFlow and/or Keras, of which 62 were from Github and 38 from Stack Overflow. This benchmark, named defect4ML, meets all standardized benchmark criteria and addresses the primary challenges of Software Reliability Engineering challenges in ML-based systems by offering bugs in various ML frameworks with different versions, comprehensive information on dependencies, and required data to trigger the bug, detailed information about the type of bugs, and link to the origin of the bug.

The third article focuses on using generative Deep Learning models trained on source code for supporting program repair tasks. In particular, the authors investigate the impact of different code representations on learning-based program repair. Such a study is particularly relevant and timely, as there is no clear understanding of the best source code representation for learning-based tasks among the various alternatives. The authors present a controlled experiment considering 21 different generative models to evaluate whether the automatically generated fixes consist of valid code ready to be used as a patch. Moreover, this work also evaluates the quality of the proposed fixes by involving developers, whose point of view is often neglected in similar studies. Interestingly, the authors find that a higher abstraction level increases accuracy, but is detrimental in terms of usefulness as perceived by developers. The results of this experiment indicate that there is no single best code representation applicable to all bug types, which could have practical implications, e.g., in program repair techniques using multiple code representations.

Finally, the last article of the special issue focuses on a large-scale empirical study that investigates the effect different types of attacks and faults have on Federated Learning aggregators. The authors have applied eight attacks and mutators (aiming to simulate faults) against four mainly used aggregation techniques focusing on image classification subject systems. The results demonstrate that there is no aggregator that outperforms the others in terms of robustness, as the federated learning aggregator's robustness depends on various parameters such as the dataset, the attack, and the data distribution. Therefore, the authors propose using an ensemble approach that combines different aggregation techniques together. The evaluation reveals that the ensemble approach benefits from the individual strengths of the aggregators to improve the quality of FL learning under attack and outperforms them in 75% of the cases.

Collectively, these four papers provide a detailed compilation of the diverse range of issues and solutions currently being investigated in the intersection between the ML/DL and Software Engineering fields.

Andrea Stocco, Onn Shehory, Gunel Jahangirova,
Vincenzo Riccio, Eitan Farchi, Guy Barash, Diptikalyan Saha
Guest Editors

Acknowledgements We thank the authors for their excellent manuscripts. We are deeply grateful to the reviewers of this special issue for their time and constructive feedback, which helped to shape the articles. We are also thankful to the Empirical Software Engineering journal and the Editors-in-Chief, Robert Feldt and Thomas Zimmermann, for their support throughout the process of preparing this special issue.

Funding Open Access funding enabled and organized by Projekt DEAL.