Detecting Operational Adversarial Examples for Reliable Deep Learning

Xingyu Zhao, Wei Huang, Sven Schewe, Yi Dong, Xiaowei Huang
Department of Computer Science
University of Liverpool
Liverpool, U.K.
{xingyu.zhao,w.huang23,sven.schewe,yi.dong,xiaowei.huang}@liverpool.ac.uk

***Abstract*—The utilisation of Deep Learning (DL) raises new challenges regarding its dependability in critical applications. Sound verification and validation methods are needed to assure the safe and reliable use of DL. However, state-of-the-art debug testing methods on DL that aim at detecting adversarial examples (AEs) ignore the operational profile, which statistically depicts the software's future operational use. This may lead to very modest effectiveness on improving the software's delivered reliability, as the testing budget is likely to be wasted on detecting AEs that are unrealistic or encountered very rarely in real-life operation. In this paper, we first present the novel notion of "operational AEs" which are AEs that have relatively high chance to be seen in future operation. Then an initial design of a new DL testing method to efficiently detect "operational AEs" is provided, as well as some insights on our prospective research plan.**

***Index Terms*—Deep Learning robustness, operational profile, safe AI, robustness testing, software reliability, software testing.**

I. Context and Motivation

The field of Verification and Validation (V&V) of Deep Learning (DL) software has recently enjoyed much activity and progress, with **robustness** being one of the properties, and probably *the* property, in the limelight. DL Robustness requires that the decision of the DL model is invariant against small perturbations on inputs. While definitions of robustness vary, they share the intuition that all inputs in an input region η have the same prediction label, where η is usually a small norm ball (defined in some L_p-norm distance) around an input x. Inside η, if an input x' is classified differently to x by the DL model, then x' is an *adversarial example*[1] (AE) – a "failure point" (in the input space) – which is loosely called a "bug" by the machine learning community.

All V&V methods for DL robustness are essentially about *detecting* AEs. There are emerging studies on systematically evaluating the AE detection ability of state-of-the-art V&V methods, e.g. [1]. One of the criteria is *naturalness*, for which the work [1] introduces quantitative metrics to assess how *realistic* the generated test cases are. Because fixing the AEs detected by realistic/natural inputs would have more practical impact on the DL model's dependability. Indeed, testing DL models with natural/realistic inputs is neither new nor against common sense; DeepXplore [2], e.g., uses domain-specific constraints to generate test cases that are valid and realistic.

However, the **delivered reliability** as a *user-centred* property [3] requires more than just detecting natural/realistic bugs. A vivid example in [4] shows that it would take 5,000 years of execution (based on users' day-to-day operation) to reveal about one third of the bugs in some tradition software systems. Clearly, spending all testing budget on detecting those "5,000 years bugs" is unwise. This is also true for DL software: given a limited testing budget, to reveal as many (potentially rare) AEs as possible is misleading. We therefore have to focus on areas in the input space that are more likely to be executed by users: we want to detect AEs from the high density area of the *Operational Profile* (OP) – a probability distribution defined over the whole input domain quantifying how the software will be operated [5].

Remarkably, our new "operational AEs" concept is more stringent than realistic/natural AEs [6]: operational AEs are realistic/natural, but not vice versa. While studies on detecting realistic/natural AEs emerge, there is no dedicated techniques for detecting operational AEs, which motivates this research.

II. Research Objectives

Our main objective is to design and implement a new "debug" testing method for DL that ensures the OP information is explicitly considered, so that the detected AEs have practical impact on the delivered reliability. Specifically, compared to the state-of-the-art DL testing methods, we aim to integrate and optimise the trad-off between the following information:

a) The OP, which is not necessarily the same as the distribution of existing data (the training and testing datasets) nor constant after deployment.

b) Naturalness. Ideally the OP returns the probability of being executed for each point in the input space. However, practically we might not have a sound OP estimator at the very fine-grain level for every single input, rather a coarse-grain level for a "cell" of inputs (e.g. a small norm ball around a natural point input). Thus, as a fallback solution, we have to apply quantified naturalness as an approximation to the "local OP" inside each cell.

c) Gradient of Loss. Generating test cases faithfully according to the OP (including naturalness as an approximation of the "local OP") is categorised as *operational testing*, which is

[1]Although named as "adversarial", x' could be either a benign input with perturbations from natural environments or a malicious attack from attackers, and we confine this research to benign inputs.

known to be inefficient in detecting bugs [7]. Given a limited testing budget, e.g. a number of test cases, the gradient of loss over the input space must be incorporated to fulfil our goal of detecting as many "operational AEs" as possible.

III. Research Questions and Methodology

To achieve the aforementioned objective, we propose a five-step iterative solution, as shown in Figure 1. Each step corresponds to a research question (RQ): given a DL model and its specific application, we first learn the OP based on which an operational dataset is synthesised (RQ1). Second, we design a *weight-based sampling* algorithm (RQ2) to sample the "seeds" (of test cases) from the operational dataset. We then implement a novel fuzzing attacking algorithm guided by naturalness (RQ3) to generate test cases around each seed. Based on the AEs detected by test cases, we retrain the DL model (RQ4). Finally, we assess the reliability of the retrained DL model (RQ5), where the result indicates how to fine tune the sampling and attacking algorithms in the next loop. Steps 2 to 5 are repeated until the required reliability level is achieved.

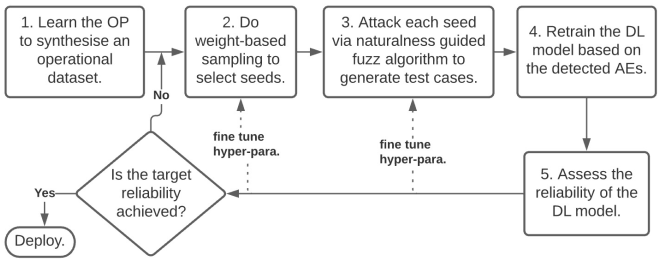
1. Learn the OP to synthesise an operational dataset.
2. Do weight-based sampling to select seeds.
3. Attack each seed via naturalness guided fuzz algorithm to generate test cases.
4. Retrain the DL model based on the detected AEs.
5. Assess the reliability of the DL model.
Is the target reliability achieved?
No
Yes
Deploy.
fine tune hyper-para.
fine tune hyper-para.

Fig. 1: Workflow of the proposed solution.

Specifically, we investigate the following RQs.

(i) RQ1: How to effectively learn the OP? It is common that the future OP and the existing dataset used for training are mismatched. Training data is normally collected in a *balanced* way, so that the DL model may perform well in all categories of input, especially when the OP is uncertain at the time of training. Given a DL application, in addition to conventional ways of building the OP [8], we envisage that techniques like high-fidelity simulation and data augmentation [9] are essential to speed up the learning and validation of the OP.

(ii) RQ2: How to select seeds that are from high density areas on the OP and also in the "buggy area" of the input space? Similar questions are initially answered in [10] by using *weight-based sampling* from the operational data set, where the weights are calculated based on auxiliary information that indicates which data-points are likely to cause failure.

(iii) RQ3: Given a seed, how to generate test cases that may efficiently detect AEs considering naturalness? Although existing attacking algorithms (e.g. [11]) perform well in efficiently detecting AEs around seeds, constraints on naturalness (local OPs) needs to be incorporated by new fuzzing algorithms.

(iv) RQ4: How to retrain the DL model based on the detected operational AEs? Adversarial training provides a potential solution to the question, yet existing methods ignore the OP information. Ideally, an enhanced adversarial training approach would consider both the OP and the detected operational AEs, while being light-weight.

(v) RQ5: How to accurately assess the delivered reliability of DL? The reliability assessment result should not only provide a stopping rule of our testing regime, but also indicate how to fine tune the sampling and attacking algorithms in RQ2 and RQ3. Our ongoing project [12] provides a preliminary assessment model [13], in which OP information and robustness evidence are considered to support reliability claims.

IV. Future Work

In our future work, we plan to develop the new DL testing tool ourlined in Figure 1 by tackling the five RQs. Although the potential methods to address each RQ are discussed, it is challenging to implement those methods and integrate them into a holistic solution. That said, conducting comprehensive and rigorous evaluation experiments on our new approach and comparing the result to state-of-the-art is crucial, not only for the calibration of our approach but also to test (and hopefully demonstrate) its superiority, e.g. requiring significant less amount of test cases to achieve the same level of reliability.

Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 956123. This work is partially supported by the UK Dstl (through the project of Safety Argument for Learning-enabled Autonomous Underwater Vehicles). We thank Lorenzo Strigini for his insightful comments.

References

[1] F. Harel-Canada, L. Wang, M. A. Gulzar, Q. Gu, and M. Kim, "Is neuron coverage a meaningful measure for testing deep neural networks?" in *Proc. of the 28th ACM Joint Meeting on ESEC/FSE*, 2020, pp. 851–862.
[2] K. Pei, Y. Cao, J. Yang, and S. Jana, "DeepXplore: Automated whitebox testing of Deep Learning systems," in *Proc. of the 26th Symposium on Operating Systems Principles*, ser. SOSP '17. ACM, 2017, pp. 1–18.
[3] B. Littlewood and L. Strigini, "Software reliability and dependability: A roadmap," in *Proc. of the Conference on The Future of Software Engineering*, ser. ICSE '00. ACM, 2000, pp. 175–188.
[4] E. N. Adams, "Optimizing preventive service of software products," *IBM Journal of Research and Development*, vol. 28, no. 1, pp. 2–14, 1984.
[5] J. D. Musa, "Operational profiles in software-reliability engineering," *IEEE Software*, vol. 10, no. 2, pp. 14–32, 1993.
[6] Z. Zhao, D. Dua, and S. Singh, "Generating natural adversarial examples," in *International Conference on Learning Representations*, 2018.
[7] P. G. Frankl, R. G. Hamlet, B. Littlewood, and L. Strigini, "Evaluating testing methods by delivered reliability [software]," *IEEE Transactions on Software Engineering*, vol. 24, no. 8, pp. 586–601, 1998.
[8] L. Strigini and B. Littlewood, "Guidelines for Statistical Testing," Tech. Rep., 1997. [Online]. Available: http://openaccess.city.ac.uk/254/
[9] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "DeepRoad: GAN-Based metamorphic testing and input validation framework for autonomous driving systems," in *ASE'18*. ACM, 2018, pp. 132–142.
[10] A. Guerriero, R. Pietrantuono, and S. Russo, "Operation is the hardest teacher: estimating DNN accuracy looking for mispredictions," in *ICSE'21*, Madrid, Spain, 2021, in press.
[11] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *ICLR'18*, 2018.
[12] X. Zhao, A. Banks, J. Sharp, V. Robu, D. Flynn, M. Fisher, and X. Huang, "A safety framework for critical systems utilising deep neural networks," in *SafeComp'20*, ser. LNCS, vol. 12234, 2020, pp. 244–259.
[13] [Online]. Available: https://github.com/havelhuang/ReAsDL