Toward Testing Deep Learning Library via Model Fuzzing

Wei Kong [1] **Huayang Cao** [*1] **Tong Wang** [1] **Yuanping Nie** [1] **Hu Li** [1] **Xiaohui Kuang** [1]

Abstract

The increasing adoption of deep learning (DL) technologies in safety-critical industries has brought about a corresponding rise in security challenges. While the security of DL frameworks (Tensorflow, Pytorch, PaddlePaddle), which serve as the foundation of various DL models, has not garnered the attention they rightfully deserve. The vulnerabilities of DL frameworks can cause significant security risks such as model reliability and data leakage. In this research project, we address this challenge by employing a specifically designed model fuzzing method. Firstly, we generate diverse models to test library implementations in the training and prediction phases by optimized mutation strategies. Furthermore, we consider the seed performance score including coverage, discovery time, and mutation numbers to prioritize the selection of model seeds. Our algorithm also selects the optimal mutation strategy based on heuristics to expand inconsistencies. Finally, to evaluate the effectiveness of our scheme, we implement our test framework and conduct the experiment on existing DL frameworks. The preliminary results demonstrate that this is a promising direction.

1. Introduction

Deep learning technologies have achieved remarkable performance in various domains such as computer vision and automatic driving. As the core technology of artificial intelligence, deep learning is being rapidly adopted across industries and domains. In order to ensure the security of the entire deep learning lifecycle requires security testing to expose vulnerabilities in DL systems and improve their trustworthiness. However, current researches focus primarily on DNN model testing and test suite generation. Open-source

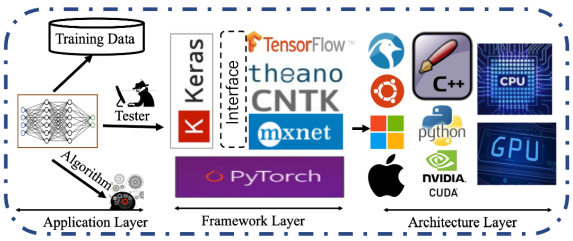

Figure 1. The architecture of Deep Learning System

DL frameworks, as a toolkit for developing AI algorithms and implementing applications, have not garnered the attention they rightfully deserve. More specifically, the vulnerabilities from DL frameworks themselves can lead to a range of security issues including data leakage and model reliability. In addition, researchers also demonstrated the ability to implant backdoors in models through attacks on the training code within the framework (Bagdasaryan & Shmatikov, 2021). Fig. 1 shows the interaction between DL model layer, frameworks layer, and architecture layer in a whole perspective. Hence, to mitigate the external and internal risks associated with DL framework vulnerabilities, it is essential to establish a systematic testing paradigm to assure the security of DL system.

In 2018, Xiao et al (Xiao et al., 2018) and Zhang et al (Zhang et al., 2018) conducted in-depth studies on three deep learning frameworks, Caffe, TensorFlow, and Torch, revealing the dependency complexity of popular deep learning frameworks and highlighting the existence of multiple vulnerabilities in these frameworks. In 2019, Pham et al (Pham et al., 2019) proposed using pre-trained models as inputs to invoke deep learning libraries and capture run-time triggered inconsistencies through differential testing. However, this approach relies on existing pre-trained models, primarily targeting common tasks, and thus can only trigger a small fraction of errors in the framework. Building upon this approach, Guo et al (Guo et al., 2020) proposed a search-based strategy for varying API parameters, weights, and input data to generate more API parameter values and more complex test samples, but Audee could only mutate the collected pre-trained DNN models. Furthermore, Wang et al (Wang et al., 2020) proposed Lemon to design a series of mutation strategies for deep learning models to explore

[*]Equal contribution [1]National Key Laboratory of Science and Technology on Formation System Security, Beijing, China. Correspondence to: Xiaohui Kuang <xiaohui$_{k}uang$@163.com>.

2^{nd} AdvML Frontiers workshop at 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

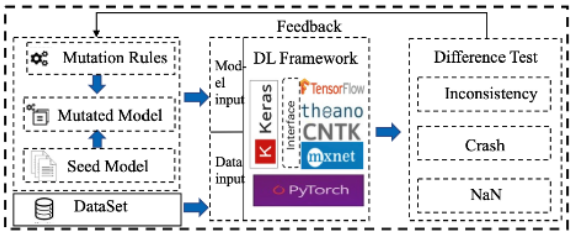

Figure 2. Workflow of our scheme

the different call sequences and hard-to-trigger behaviors of deep learning library. However, this scheme can be further improved on the model mutation strategies to test more types of framework vulnerabilities. Additionally, Lemon can only mutate the model to test the vulnerability of the DL framework in the prediction phase without considering the model training phase.

Existing model mutation strategies focus on generic intra-layer and inter-layer neuron weights and positions, without taking into consideration of structural mutations within the neural network, such as activation functions, and layer structures. However, existing DL testing schemes utilize pre-trained models as seeds for mutation, thereby reducing the model training time overhead. As a result, it is difficult to test the library implementations used during the training process because the pre-trained models no longer require the training process.

2. Research Contents

To address the aforementioned issues, this work aims to optimize model mutation strategies, generate a more diverse set of seed models, and apply mutation strategies to the retraining process. Fig. 2 describes the procedure of testing.

1) Model mutation strategy optimization: We seek to improve typical model mutation strategies, such as weight mutations, neuron activation state changes, and neuron location alterations. Building upon these strategies, we define mutation strategies at the source level and design a heuristic mutation strategy selection algorithm to increase the degree of output inconsistency across different frameworks.

2) Mutation seed selection: The selection of mutation seeds is important to test DL frameworks. For the problem that the currently proposed model mutation seed selection does not trigger the libraries in the neural network training process, this work intends to first collect existing DNN models as seeds for mutation. The mutation strategy is further applied during the model training phase to activate the underlying layers of the framework, such as Dropout and loss optimization

3. Methodologies

This work aims to generate more diverse test models that can trigger more libraries and increase the likelihood of inconsistencies or crashes on different frameworks.

1) Model mutation strategy optimization: In our work, the network structure mainly includes intra-layer and inter-layer neuron mutation strategies, including layer swapping, adding layers, deleting layers, neuron swapping, weight mutation, adding neurons, and deleting neurons. Additionally, the mutation in the neural network parameters includes the size of the neural network and the parameters related to the computation such as activation function selection, dropout, loss function, and penalty function.

2) Mutation seed selection: The process of generating mutated models is iterative, and the models generated in the previous iteration can also be used as seeds for the next iteration, especially models that can lead to a greater degree of inconsistency. To increase model diversity, a seed model that has rarely selected mutations should be given a higher priority to participate in mutations. Hence, the seed performance score of the proposed definition model includes coverage (seeds that can trigger more libraries are preferred), discovery time (seeds that are found later are preferred), and number of mutations (seeds with fewer mutations are preferred) to jointly determine the seed priority.

3) Mutation strategy selection algorithm: Our work takes a set of seed models and model mutation strategies as inputs. Next, we utilize multiple heuristics (e.g. greedy algorithm, or genetic algorithm) to find the optimal mutation strategies that expand the inconsistency. In the current result, we found that greedy-based mutation strategy selection algorithms can efficiently generate more high-priority model seeds. Finally, the mutation process is repeated in a depth-first manner to generate more test models.

4. Ongoing and Future work

This work intends to address the problem that existing model mutation-based schemes primarily rely on pre-trained models for testing. Therefore, we proposed a DL framework testing technique based on model fuzzing to generate more diverse models that can trigger more libraries in the framework. Currently, We have established DL framework testing platform based on Linux with 4*NVIDIA K40 GPU, which has integrated the mainstream DL frameworks (e.g. Tensorflow, Pytorch and Theano). Furthermore, we plan to continuously optimize our mutation strategies and to verify the optimal algorithm for selecting mutation strategies. Furthermore, we will reproduce the disclosed vulnerabilities and test the latest version of DL frameworks (i.e. PaddlePaddle) on our testing platform.

References

Bagdasaryan, E. and Shmatikov, V. Blind backdoors in deep learning models. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 1505–1521, 2021.

Guo, Q., Xie, X., Li, Y., Zhang, X., Liu, Y., Li, X., and Shen, C. Audee: Automated testing for deep learning frameworks. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, pp. 486–498, 2020.

Pham, H. V., Lutellier, T., Qi, W., and Tan, L. Cradle: cross-backend validation to detect and localize bugs in deep learning libraries. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pp. 1027–1038. IEEE, 2019.

Wang, Z., Yan, M., Chen, J., Liu, S., and Zhang, D. Deep learning library testing via effective model generation. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 788–799, 2020.

Xiao, Q., Li, K., Zhang, D., and Xu, W. Security risks in deep learning implementations. In *2018 IEEE Security and privacy workshops (SPW)*, pp. 123–128. IEEE, 2018.

Zhang, Y., Chen, Y., Cheung, S.-C., Xiong, Y., and Zhang, L. An empirical study on tensorflow program bugs. In *Proceedings of the 27th ACM SIGSOFT international symposium on software testing and analysis*, pp. 129–140, 2018.