Detection of failing tests in AI-based systems

Antonio Guerriero*, Michael R. Lyu†, Roberto Pietrantuono*, Stefano Russo*
* Università degli Studi di Napoli Federico II
{antonio.guerriero, roberto.pietrantuono, stefano.russo}@unina.it
† The Chinese University of Hong Kong
lyu@cse.cuhk.edu.hk

***Abstract*—With Artificial Intelligence (AI) being increasingly used in critical systems, reliability of AI-based systems is a great concern. Testing is a fundamental software reliability engineering technique, but its application to AI-based systems poses new challenges, as determining the correct output for an arbitrary input may be hard (*oracle problem*). Building on the notion of *failure detectors* in distributed systems, we propose a surrogate of a test oracle (*Failed Test Detector*, FTD) able to unveil when the output of an AI-based system is incorrect despite the correct output is uncertain. An FTD should conveniently trade off *completeness* and *accuracy*.**

***Index Terms*—Testing, Artificial Intelligence, Reliability**

I. Introduction

Testing is fundamental in software reliability engineering [1], yet testers are facing new challenges with the increasing use of Artificial Intelligence (AI) in software systems [2]. Testing aims to expose failures of the system under test, and a *test oracle* is needed to establish if a test "passes" or "fails". However an oracle may be hard to define for AI-based systems, due to possible uncertainty on their correct output [3].

This is attributable to the software development paradigm shift inherent to Machine Learning (ML) [4] and AI, from definition and coding of deterministic algorithms to training and learning from data with statistics algorithms. Most traditional testing techniques foresee the specification of input test data and context conditions and the *a priori* identification of the expected output. For an AI-based system, an objective and consistent specification, against which the system behavior can be tested, is rarely available, due to the uncertainty in system output for some input data, and for the same test data in varying context conditions [3]. As stated in [4], *generating reliable test oracle is sometimes infeasible for some ML systems*, and, according to Murphy *et al.*, more in general in AI-based systems *there is no reliable "test oracle" to indicate what the correct output should be for arbitrary input* [5].

We propose to address the *oracle problem* for AI-based systems by somehow "restricting" it to the definition of an oracle able to evaluate when a test *fails*, in situations where the exact output for that test is unknown or uncertain. Such an oracle is similar to a *failure detector* for processes in a distributed system [6]; we call it *Failed Tests Detector* (FTD).

II. AI-based systems

As stated in [3], *AI-based software and applications use machine learning models and techniques through large-scale data training to implement diverse artificial intelligent features and capabilities*. We regard an AI-based system (Figure 1) as taking a feature vector (f_i) as input; an internal component uses an AI to compute a prediction producing a response (r), while other components (c_i), not based on AI, produce additional outputs (a_i) which, combined with r, yield the ultimate output o. For instance, an autonomous driving system exploits an AI to process the cameras' images, while other components process on-board sensors' data, ultimately determining a final output (speed or steering angle). The behavior of an AI-based system is strongly dependent on both the AI algorithms and the training data.

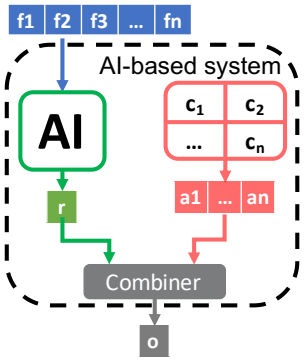

Fig. 1. AI-based system

III. Detection of failing tests

In partial analogy with distributed failure detectors [6], the output of an FTD for a given test is *fail* when the FTD is ("reasonably") able to tell that the system has produced an incorrect output for the test case, otherwise it is *unknown*. The effectiveness of an FTD can be evaluated based on the following quality metrics:

- *Completeness*: all incorrect outputs of the system under test are correctly judged as *fail* from the FTD (the FTD is complete if it is able to avoid False Negatives);
- *Accuracy*: all correct outputs of the system under test are not judged as *fail* from the FTD (the FTD is accurate if it is able to avoid False Positives).

We envisage the architecture of a Failed Tests Detector as depicted in Figure 2. It is meant to exploit various ways to discover failed tests, using different sources of knowledge (domain knowledge, training set, system's internal parameter values) to look for conditions, e.g. *invariants*, which hold when the system output can be judged incorrect, despite the

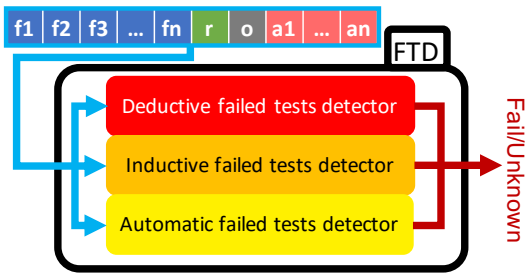

Fig. 2. Architecture of a Failed Tests Detector

correct output is unknown or uncertain. Consider, for instance, the mentioned autonomous driving system. Its output o, e.g. steering angle and speed, depends on the response r of the AI component that processes the acquired images, and on the outputs (a_i) of on-board sensors. The architecture we envisage foresees three methods by means of which the FTD can detect failed tests:

- *Deductive method*: it is based on rules defining domain-related conditions that should never be violated (e.g., driving rules) – they, in a sense, describe the domain (we call them *domain invariants*). They can be derived from an expert, from an ontology, or via deductive processing of other rules. Each output violating such rules is judged as a failure.
  This level is characterized by a deterministic evaluation of the output, detecting a subset of failed tests with 100% accuracy, but incomplete (i.e., only a subset of failures is identified, depending on the set of rules being defined) – hence with false negatives with respect to the set of all possible failures, but with no false positives.
  An implementation of this method is called *Deductive FTD*. A deductive rule to detect failures for the example is "*if (the proximity alarm is ON* $\wedge$ *speed* > 0.1 *Km/h)* $\Rightarrow$ *fail*".
- *Inductive method applied to training data*: a set of rules is inferred from the prior knowledge in the training set. The rules should be verifiable by the tester and able to identify a subset of failures preserving high accuracy (depending on the goodness of the training data). These are "likely" invariants (we call them *data-dependent invariants)* that we expect, with a certain probability, to observe upon a failure occurrence. Adding this level, the completeness can be significantly increased, depending on the representativeness of the training data.
  An implementation of this method is called *Inductive FTD*. An inductive rule for the example is "*if* $(pixel_{12,35} > 200 \wedge pixel_{14,65} < 56 \wedge speed > 75$ *Km/h)* $\Rightarrow$ *fail*".
- *Inductive method applied to AI-internal data*: the aim is to mine patterns about how the AI produces an incorrect response. The assumption is that failures have similar patterns that are not available in the training set, and that are specific to the adopted algorithm. Hence we look for likely invariants in the AI behavior (*AI invariants*). This solution relies on the possibility to generate a dataset able to show failures of the system under test, to derive the invariants. Adding this level, we expect to improve the completeness of the FTD detecting further different kinds of failures, potentially decreasing its accuracy (the number of false positives could increase).
  An implementation of this method is called *Automatic FTD*. An inductive rule for the example, assuming that the AI is implemented considering a Neural Network, is "*if (a certain subset of neurons has been activated* $\wedge$ *speed* > 40 *Km/h)* $\Rightarrow$ *fail*".

As a (contrived) example where the correct output is uncertain, consider an autonomous car finding on its road an unexpected obstacle (Figure 3). The AI has to select the steering angle starting from the image captured from the frontal camera. Assume the obstacle hides the road signs, and the system output is "turn left". An FTD which can take into account additional information (e.g., from a GPS navigator), telling that the street on the left is one-way, can judge the system decision as a failure: even if the collision is avoided, the action performed is likely to produce an accident.

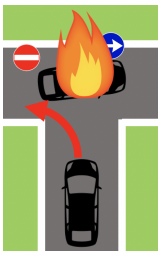

Fig. 3. A scenario with uncertain correct output for an autonomous car

IV. Conclusions

To tackle the *oracle problem* in testing AI-based systems, we propose to borrow the notion of *failure detectors* for reliable distributed systems, so as to establish when the output of a system is incorrect even when the correct output is uncertain. We therefore introduce a Failed Tests Detector, and suggest it can be realized using deductive and inductive reasoning. Currently, we are running experiments on the MNIST dataset with a prototypal FTD. Preliminary results show that a trade-off between FTD completeness and accuracy can be achieved, providing useful support to testers of AI-based systems.

References

[1] M. R. Lyu, Ed., *Handbook of software reliability engineering*. Hightstown, NJ, USA: McGraw-Hill, Inc., 1996.
[2] D. Marijan, A. Gotlieb, and M. K. Ahuja, "Challenges of Testing Machine Learning Based Systems," in *IEEE Int. Conf. on Artificial Intelligence Testing (AITest)*. IEEE, 2019, pp. 101–102.
[3] J. Gao, C. Tao, D. Jie, and S. Lu, "Invited paper: What is ai software testing? and why," in *Proc. IEEE Int. Conf. on Service-Oriented System Engineering (SOSE)*. IEEE, 2019, pp. 27–46.
[4] Z. Wan, X. Xia, D. Lo, and G. C. Murphy, "How does Machine Learning Change Software Development Practices?" *IEEE Trans. on Software Engineering*, Available online: 26 August 2019.
[5] C. Murphy, G. E. Kaiser, and M. Arias, "An approach to software testing of machine learning applications," in *Proc. 19th Int. Conf. on Software Engineering & Knowledge Engineering (SEKE)*, 2007, pp. 167–172.
[6] T. D. Chandra and S. Toueg, "Unreliable failure detectors for reliable distributed systems," *J. of the ACM*, vol. 43, no. 2, pp. 225–267, 1996.