Towards Simplification of Failure Scenarios for Machine Learning-enabled Autonomous Systems

Donghwan Shin*, Sanjeetha Pennada
University of Sheffield, Sheffield, United Kingdom
d.shin@sheffield.ac.uk, s.pennada@sheffield.ac.uk
*Corresponding author

Abstract—Scenario-based testing is an essential way of improving the safety and reliability of machine learning-enabled autonomous systems (MLAS), such as autonomous driving systems (ADS). As the complexity of failure scenarios increases with the development of more realistic MLAS testing approaches, it becomes essential to simplify failure scenarios to understand and identify the root causes of failures. In this vision paper, we present our vision to leverage search-based software engineering (SBSE) and surrogate-assisted optimisation (SAO) to address the challenges of simplifying failure scenarios in MLAS.

Keywords–Autonomous Systems, Software Testing, Scenario Simplification

1. Introduction

An autonomous system is an intelligent system (e.g., an autonomous driving system) designed to perceive and act on the surrounding physical environment (e.g., a highway driving environment, including other traffic participants) and work for extended periods of time without human intervention [1]. To ensure the safety and reliability of the autonomous behaviour of the system, therefore, it is essential to consider a *scenario*. For example, in the automotive domain, a scenario can be defined as a set of static entities (e.g., trees, buildings, traffic signs, and other side objects) and dynamic entities (e.g., moving vehicles, pedestrians, traffic lights, and other state-changing objects) that might affect the behaviour of autonomous driving systems (ADS). By including all the details (e.g., positions, sizes, shapes, and trajectories) of the static and dynamic entities required to recreate the situation, scenarios become arguably the most important artefacts to verify and validate MLAS [2].

Many approaches have been proposed in the literature to automatically generate failure scenarios with the aim of revealing the failures of MLAS in various domains. One common research direction of these *failure scenario* generation studies is the increasing complexity of scenarios to capture more realistic situations. For example, early ADS testing studies focused on a few environmental factors (e.g., weather conditions, and road topologies), whereas recent studies start to cover more diverse factors (e.g., all controllable factors in a driving simulator with various driving obstacles) in a single scenario. Together with the increasing fidelity of driving simulations [3], the complexity of failure scenarios leads to an essential question: *Given a complex failure scenario, what is the minimum set of scenario entities required to cause the same failure?*

In this vision paper, we define the problem of MLAS failure scenario simplification with the major challenges to address (Section 2) and propose a research vision to tackle the challenges (Section 3).

This research is supported by the Engineering and Physical Sciences Research Council [grant number EP/Y014219/1].

2. Problem Definition

The definition of the MLAS failure scenario simplification problem is as follows.

Definition 1 (MLAS Failure Scenario Simplification)**.** For a given MLAS M, its failure scenario S that includes a set of scenario entities, and the failure F of M induced by S, the problem of MLAS failure scenario simplification is to identify a simplified failure scenario S' that includes the minimum set of scenario entities in S while inducing F.

Addressing the problem entails four main challenges. First, due to the non-linear nature of ML models and their interactions in MLAS, a failure scenario might have different (possibly disjoint) sets of entities inducing the same failure. This prevents us from using conventional techniques to minimise test inputs [4, 5] as they do not consider multiple sets of failure-inducing entities to identify failure-inducing entities efficiently based on assumptions about monotony and unambiguity [5].

Second, since multiple set of entities include the same failure in a given failure scenario, potential solution candidates consist of all possible sets of entities, the number of which is exponential in the number of scenario entities. This makes it impossible to exhaustively explore the entire problem space to find the minimum inducing the same failure.

Third, MLAS often contain components (in particular ML models) developed by 3rd parties [6]. Therefore, we cannot assume the availability of the source code, documentation, or other internal information of MLAS.

Fourth, whereas conventional techniques can take advantage of many software system executions, this is out of the question for MLAS, where a single execution can take several minutes

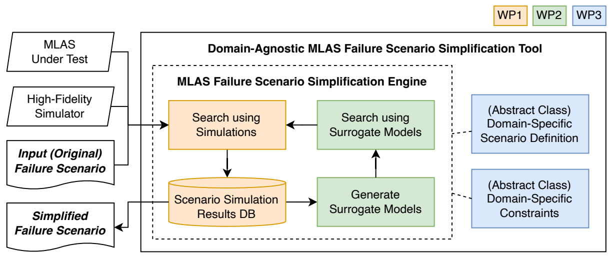
WP1
WP2
WP3
MLAS Under Test
High-Fidelity Simulator
Input (Original) Failure Scenario
Simplified Failure Scenario
Domain-Agnostic MLAS Failure Scenario Simplification Tool
MLAS Failure Scenario Simplification Engine
Search using Simulations
Search using Surrogate Models
Scenario Simulation Results DB
Generate Surrogate Models
(Abstract Class) Domain-Specific Scenario Definition
(Abstract Class) Domain-Specific Constraints

Figure 1. Overview of the proposed approach

if not hours due to the involvement of computationally expensive, high-fidelity simulations [3].

3. Research Vision

We envision that we can address the problem by carefully combining search-based software engineering (SBSE) [7] and surrogate-assisted optimisation (SAO) [8]. SBSE is a well-established research field in which a target problem (e.g., test data generation) with too many possible solutions to be completely enumerated or explored is formulated as an optimisation problem and solved using metaheuristics. SAO is an optimisation approach that effectively utilises computationally lightweight surrogate models instead of computationally intensive simulations to minimise fitness evaluation costs in metaheuristics. To achieve the vision, we plan to conduct three work packages (WPs), as illustrated in Figure 1.

WP1: Search-based simplification. We start with converting the failure scenario simplification problem into an optimisation problem by focusing on two objectives: automatically simplify a given MLAS failure scenario as much as possible while inducing the same failure without any further information about MLAS. We will define a representation (encoding) of a candidate solution (i.e., a scenario simplified from an MLAS failure scenario), a fitness function that guides the search towards simpler scenarios while inducing the same failure, and search operations (e.g., selection, mutation, and crossover) to balance exploration and exploitation of the search space [7].

WP2: Surrogate-assisted simplification. On top of the outcome of WP1, we will address the challenge of computationally intensive simulations required for fitness evaluations using SAO. We will define surrogate models that can predict the probability of inducing the same failure for a given scenario without simulations. We will then develop a surrogate-assisted search technique, following the state-of-the-art [6].

WP3: Domain-agnostic extension. For WP1 and WP2, we will mainly focus on the automotive domain, where open-source MLAS and high-fidelity simulators are readily available [6]. In WP3, however, we will extend the proposed approach to other domains beyond ADS, such as mobile robots and unmanned aerial vehicles, where the minimisation of failure scenarios is essential. With more collaborations with domain experts (e.g., Sheffield Robotics and Advanced Manufacturing Research Centre), we will generalise ADS-specific components and techniques to create a domain-agnostic tool that can be applied to various MLAS.

4. Ongoing and Future Work

As the initial stage, we generated ADS failure scenarios and manually investigated them using a high-fidelity driving simulator [3] to better understand the scenario entities affecting the failure. Specifically, we considered failure scenarios consisting of various scenario entities, such as weather conditions, road topologies, buildings, traffic lights, and other (adversarial) vehicles. During this preliminary study, we observed that certain scenario entities were unexpectedly influential in causing the failure. For example, in one of the failure scenarios, simply switching weather conditions from Clear to Rainy resulted in avoiding the failure (collision), whereas removing roadside buildings did not affect the failure. However, we do not know what is the true minimum set of scenario entities required to cause the same failure. Considering the sheer number of scenario entities and their interactions, automated failure scenario simplification is essential.

Future work is to (1) develop a search-based simplification technique, including a new representation of a (simplified) scenario, a fitness function, and search operations, and (2) develop surrogate models to reduce the cost of the search-based simplification.

References

[1] H. Araujo, M. R. Mousavi, and M. Varshosaz, "Testing, validation, and verification of robotic and autonomous systems: A systematic review," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 2, mar 2023.

[2] S. Riedmaier, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer, "Survey on scenario-based safety assessment of automated vehicles," *IEEE Access*, vol. 8, pp. 87 456–87 477, 2020.

[3] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 78. PMLR, 13–15 Nov 2017, pp. 1–16.

[4] A. Zeller and R. Hildebrandt, "Simplifying and isolating failure-inducing input," *IEEE Transactions on Software Engineering*, vol. 28, no. 2, pp. 183–200, 2002.

[5] G. Wang, R. Shen, J. Chen, Y. Xiong, and L. Zhang, "Probabilistic delta debugging," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, 2021, p. 881–892.

[6] F. U. Haq, D. Shin, and L. Briand, "Efficient online testing for dnn-enabled systems using surrogate-assisted and many-objective optimization," in *Proceedings of the 44th International Conference on Software Engineering*. ACM, 2022, p. 811–822.

[7] M. Harman, S. A. Mansouri, and Y. Zhang, "Search-based software engineering: Trends, techniques and applications," *ACM Comput. Surv.*, vol. 45, no. 1, dec 2012.

[8] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 61–70, 2011.