Risk-driven Online Testing and Test Case Diversity Analysis for ML-enabled Critical Systems

Jubril Gbolahan Adigun*⊥♠, Tom Philip Huck¶, Matteo Camilli†, Michael Felderer*‡§
* University of Innsbruck, Austria
Email: {first}.{last}@uibk.ac.at
⊥ Ainnov8 Technologies Ltd, Nigeria
Email: jubril@ainnov8.com
♠ Center for Artificial Intelligence (AI) Research Nepal, Nepal
Email: jubril.adigun@cair-nepal.org
¶ Karlsruhe Institute of Technology (KIT), Germany
Email: tom.huck@kit.edu
† Politecnico di Milano, Italy
Email: matteo.camilli@polimi.it
‡ German Aerospace Center (DLR), Germany
Email: michael.felderer@dlr.de
§ University of Cologne, Germany
Email: michael.felderer@uni-koeln.de

***Abstract*—Machine Learning (ML)-enabled systems that run in safety-critical settings expose humans to risks. Hence, it is important to build such systems with strong assurances for domain-specific safety requirements. Simulation as well as metaheuristic optimizing search have proven to be valuable tools for online testing of ML-enabled systems for early detection of hazards. However, the efficient generation of effective test cases remains a challenging issue. In particular, the testing process shall produce as many failures as possible but also unveil diverse sets of failure scenarios.**

To study this phenomenon, we introduce a risk-driven test case generation and diversity analysis method tailored to ML-enabled systems. Our approach uses an online testing technique based on metaheuristic optimizing search to falsify domain-specific safety requirements. All test cases leading to hazards are then analyzed to assess their diversity by using clustering and interpretable ML. We evaluated our approach in a collaborative robotics case study showing that generating tests considering risk metrics represents an effective strategy. Furthermore, we compare alternative optimizing search algorithms and rank them based on the overall diversity of the test cases, ultimately showing that selecting the testing strategy based on the number of failures only may be misleading.

***Index Terms*—Search-based testing, ML-enabled systems, Risk, Diversity analysis, Simulation**

I. Introduction

Autonomous systems running in safety-critical settings (e.g., collaborative robots, autonomous vehicles) increasingly rely on machine learning (ML) components (e.g., Deep Neural Networks) to mimic aspects of human intelligence, such as vision-related tasks (e.g., image classification, and object detection). Collaborative robots, sometimes referred to as *Collaborative Artificial Intelligence Systems* (CAISs) [1], [2], belong to this class. The core functions of these systems are enabled by ML components to work together with humans in a shared physical space and achieve a common goal. Indeed, the internal logic of ML components is not rigorously specified and is not captured by a piece of code but is rather determined by a training process involving positive and negative examples (i.e., pre-labeled data points). The critical setting of CAISs yields hazardous circumstances that could harm human beings necessitating the need for strong assurances of compliance with domain-specific requirements.

Recent empirical studies found offline testing is less effective in uncovering safety violations compared to online testing [3]. Clearly, even if the ML components are reliable, they are embedded in complex and dynamic operational ecosystems affected by sources of uncertainty that may lead to unsafe component interactions and, therefore, to accidents [4], [5]. Online testing methods for ML-enabled systems deal with these issues by considering the ML components in a closed loop with the other parts and the surrounding environment. However, there are still open challenges in cost-effective identification of safety hazards. The input space of ML components is very large and high-dimensional. Thus, testing effort should be carefully directed toward those semantically meaningful scenarios that increase the likelihood of observing *diverse* sets of hazards (i.e., violation of safety requirements). In reality, failure diversity in ML-enabled systems is less studied compared to the ability to produce a lot of failures [6].

To address the aforementioned challenges, we propose a risk-driven online testing method and test case diversity analysis for ML-enabled systems running in safety-critical domains. Our method extracts safety requirements from domain-specific standards regulating the risk management process. Then, we recast the problem of online testing as a single-objective optimization problem that drives the generation of test cases to falsify a given safety requirement. To study the diversity of the generated test cases, our method combines unsupervised *clustering* [7] and model-agnostic *interpretable* ML [8]. The idea is to group together test cases with small distance and

then validate whether such test cases are similar in a relevant way, that is, they yield the *same hazard(s) for the same root cause(s)*, according to local explanations.

We conducted an empirical evaluation of our approach by using a CAIS case study simulated using the industry-strength robot simulator CoppeliaSim [9]. Our experiments aim at studying: (1) the effectiveness of the risk-driven test case generation across different search strategies and the statistical significance of the results; (2) the diversity of the generated test cases causing hazards. We found that selecting a particular testing strategy based on failure-revealing ability only may be misleading. Furthermore, discarding similar test cases may lead to overlooking significant differences in the testing outcome due to the non-linear, non-convex behavior of ML components.

The main contributions of this work are summarized as follows:

- a novel risk-driven online testing method that falsifies domain-specific safety requirements for collaborative robots;
- a novel test case diversity analysis method based on clustering and interpretable ML;
- the application of our methods to a CAIS case study involving an industry-strength simulator (the implementation of the methods as well as the case study artefacts are provided in our replication package);
- an empirical evaluation focusing on effectiveness, statistical significance, and test case diversity.

The remainder of the paper is as follows. In Sec. II, we introduce relevant background concepts. In Sec. III, we introduce our case study in the CAIS class of systems. In Sec. IV, we describe our novel risk-driven online testing and test case diversity analysis method, while we discuss the empirical evaluation in Sec. V. In Sec. VI, we summarize related work and we draw our conclusion in Sec. VII.

II. Preliminaries

A. Search-based Testing

Search-based testing (SBT) is an automated testing technique that formulates the testing problem as an optimization challenge by defining a proper fitness function according to an objective of the problem at hand. SBT adopts metaheuristic optimizing search algorithms [10](either single-objective or multi-objective) to automatically generate large amounts of inputs with the aim of steering the search process towards better test cases, where "better" is defined by the fitness function. According to the application domain and the testing objective, the fitness function used to evaluate the quality of the test cases must be carefully designed, as it can greatly impact the cost-effectiveness of the search process. Since SBT can be computationally expensive for large systems having many input parameters, the fitness should steer the generation toward those semantically meaningful scenarios that increase the likelihood of hazards, especially in safety critical settings.

In this latter case, field testing is usually expensive and dangerous. Hence, simulation provides a means to control virtual environments and safely test the target system relying on simulators. Even though this latter approach can raise concerns regarding the fidelity of the simulations, it has clear advantages. It allows the virtual environments to be controlled systematically, thus increasing internal validity in establishing cause-effect relationship during the testing process. Furthermore, shared phenomena between the system and the virtual environments can be directly measured from the simulations. This represents a ground truth that can be used to deal with the well-known oracle problem in testing ML-enabled systems [11]. Such a ground truth enables the adoption of SBT solutions to perform safe and automated testing for safety-critical systems.

B. Clustering

Clustering [12] refers to an unsupervised learning task in which data points are grouped into sets such that data points in the same set (i.e., cluster) are more similar to each other (based on the notion of distance) compared to data points in other clusters. Popular cluster models are, for instance, centroid models (e.g., k-means) that represent each cluster by a single mean point, connectivity models (e.g., hierarchical clustering) that builds clusters based on distance connectivity defined on hierarchical structures, and density models (e.g., DBSCAN) that define clusters as connected dense regions in the data space. For some algorithms, such as k-means, the total number of clusters must be specified beforehand. Other algorithms, such as DBSCAN [7], do not require this prior knowledge and automatically determine the optimal number of clusters. If the data space is high dimensional, clustering (or more in general data analysis) may be computationally intractable [13]. Here, dimension reduction can be beneficial. *Dimension reduction* refers to the transformation of data points from a source high-dimensional space to a target low-dimensional space. The transformation retains meaningful properties of the original data points but it reduces the number of variables. Dimension reduction methods are commonly divided into linear and non-linear approaches. Principal Component Analysis [14] (PCA) is a popular linear technique used to increase interpretability of the data while preserving the maximum amount of information of the source high-dimensional space.

C. Interpretable Machine Learning

Interpretable (or explainable) ML [8] refers to the extraction of relevant knowledge from an ML model concerning existing relations contained in the data or learned predictive models. Model-agnostic techniques are particularly useful for *black box* models that do not explain their predictions (e.g., Neural Networks). The scope of interpretability is either *global* (i.e., holistic model interpretability) or *local* (i.e., interpretability for a single prediction). Global explanations describe the average behavior of a given model. They give a holistic view of the distribution of the target outcome (e.g., class labels) based on the features.

Local explanations, such as those produced by Local Interpretable Model-agnostic Explanation [8] (LIME), take into

account an instance of interest and examine the prediction to explain possible reasons based on an interpretable surrogate model. The LIME method starts from a data point x_i and generates a new dataset consisting of perturbed samples mapping to the corresponding predictions of the original model. LIME uses the new dataset to train an interpretable model, which is weighted by the proximity of the samples to x_i. having the *local fidelity* property (i.e., it is a good approximation of local predictions, but not necessarily a good global approximation). LIME explains a given prediction in terms of values assumed by the most relevant features for a given instance. The *relevance* of features is expressed by LIMES through a mapping from features to *weight*s in $[0, 1]$. This information is used to rank the features and explain the extent to which they influence the predicted outcome. Notice that LIME explanations of different instances yield different rankings since weights depend on feature values of the instances.

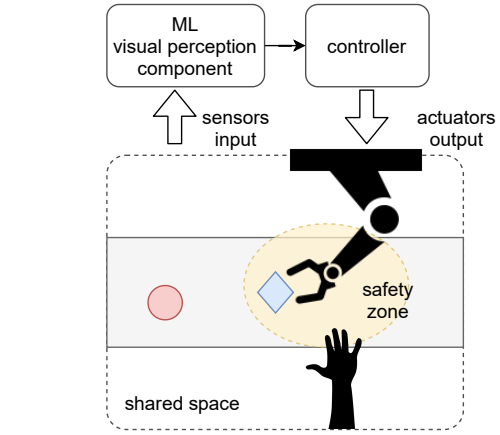
III. Collaborative AI System Case Study

Here, we introduce our case study used to illustrate the main steps of our approach, but also as a system subject in our empirical evaluation. The case study is an industrial CAIS used to carry out a production-relevant collaborative "pick and place" task along with a human operator [1], [15]. Figure 1a illustrates a high-level schema of the collaborative task, whereas Fig. 1b shows a translation of the schema to our simulated case study. In this example, an automated controller of a robotic arm attempts to detect and classify objects (e.g., by color and shape) on a conveyor belt and actuates the proper movements to pick and move the object into the right bucket. The sorting skill is acquired by the robot automatically, based on human demonstrations. The system includes a controller, an actuated mechanical system (i.e., robotic arm), and a camera sensor along with a visual perception ML component for classification. This ML component learns on structured heterogeneous data sources associated with features (e.g., shape and color of an object) and yields category labels as output, where labels identify the buckets where the objects must be placed. The operator collaborates with the robot in order to supervise the correct transfer of the desired sorting skill to the robot and can intervene through gestures when corrections are required.

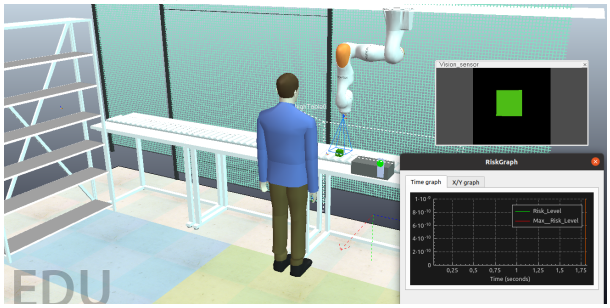
Risk management processes in the development of these industrial robots are regulated by existing standard documents (i.e., ISO 10218-1[1] and ISO 10218-2[2]). In particular the ISO/TS 15066[3] for collaborative robotics define risk mitigation procedures involving the assurance of relevant safety requirements depending on the adopted operating mode as reported in Table I. The safety control approach in our case study is implemented as described in [16], by considering the SRMS operating mode according to Table I. Here, a protective separation distance between the human and the robot is checked online using *safety zone*s [16]. The dimension of such zones is dynamically adapted based on the robot motion. Fast motions of the robot can generate large safety zones which may negatively affect the collaboration.

TABLE I: Operating modes defined by ISO/TS 15066.

Mode	Description
Power and Force Limiting (PFL)	Humans and robots can act simultaneously within the shared space and, even if they come into contact the human is unharmed since the robot is designed with rounded edges and limited force to soften potential impacts.
Safety-rated Monitored Stop (SRMS)	Robots shall always keep a protective distance (i.e., the safety zone) and interrupt any ongoing motion in case humans are too close (i.e., the human brakes the safety zone) since the robots can potentially harm them.
Hand Guidance (HG)	Contacts in collaborative tasks may occur since humans have an active role in monitoring, supervising, and controlling the robots. In this case, the robots shall dynamically activate and deactivate autonomous behaviors.



(a) Collaborative task schema.



(b) Case study simulation in CoppeliaSim.

Fig. 1: Illustrative examples of our CAIS case study.

Figure 2 illustrates a UML Activity Diagram describing the scenario of interest of our case study. The scenario includes two agents, human and robot, described in Fig. 2 using two swim lanes. The agents collaborate in the aforementioned "pick and place" task. The scenario starts with an object (a small box) placed onto the conveyor belt placed in the shared physical space. Both the human and the robot can move to pick the object up and place it into a bucket. The two agents decide when they start moving (after a certain waiting time). We

[1] https://www.iso.org/obp/ui/#iso:std:iso:10218:-1:ed-2:v1:en
[2] https://www.iso.org/obp/ui/#iso:std:iso:10218:-2:ed-1:v1:en
[3] https://www.iso.org/obp/ui/#iso:std:iso:ts:15066:ed-1:v1:en

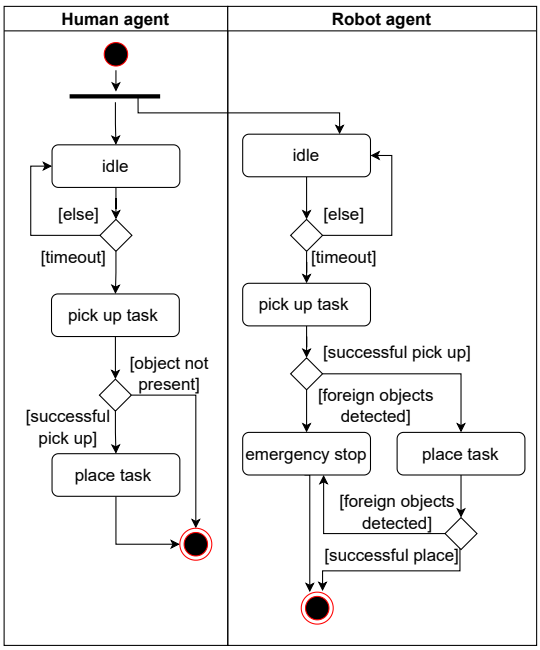


Fig. 2: UML Activity Diagram of the simulated scenario.

assume that the human does not care about possible hazards and, once started he/she tries to complete the task, no matter what is the behavior of the robotic arm. On the contrary, the robotic arm has a controller that shall trigger an emergency stop according to the SRMS operating mode. The system is equipped with a point-of-view vision sensor that feeds the pre-trained ML component in charge of detecting possible foreign objects (in this case, parts of the human agent, such as arms and hands) to trigger an emergency stop action and keep the protective distance. The scenario terminates either when: one of the two agents safely completes the task (i.e., the object has been placed into the bucket); or a hazard occurs (i.e., the protective distance is violated and the robot is moving).

The simulation of our case study is implemented using CoppeliaSim [9] (formerly known as V-REP), a high-fidelity robot simulator. Figure 1b shows a screenshot of our simulated case study running onto CoppeliaSim. The simulator is connected to an external controller implemented in Python that receives a stream of images from the vision sensor and it triggers the emergency stop actions based on the outcome of a pre-trained ML visual perception component implemented using the OpenCV vision library [17].

IV. Testing and Diversity Analysis

In this section, we first define our problem statement and then we present our approach based on risk-driven online testing and test case diversity analysis.

A. Problem Statement

In the following, we provide the reader with a general treatment of the problem regarding test case generation, test case execution, and test case diversity analysis for ML-enabled critical systems. Even though we use a specific case study in the CAISs class, in this description, we try to be as general as possible to potentially include other categories of ML-enabled critical systems that embed ML visual perception components, such as autonomous vehicles.

In our scenario of interest, the ML component takes as input a point-of-view image of the shared space (from the perspective of the robot agent) and triggers an emergency stop as soon as foreign objects are detected. The ML component is trained considering a low-level feature space (i.e., pixels sensed by the vision sensor). These features can be manipulated in a semantically meaningful way (i.e., in a realistic way avoiding adversarial perturbations) by considering higher level (and human interpretable) factors of interest, such as the position and other physical properties of the human operator, the position of the vision and other relevant factors affecting the field of view, thus impacting the quality of the data sensed from the environment. By monitoring the running scenario, we can detect hazards according to the violation of safety requirements extracted from relevant domain-specific regulatory documents. In our case, we consider the ISO TS 15066, and in particular the SRMS operating mode. Given the function $\mathcal{S}$ that computes the boundary of the *safety zone* [16] according to robot location and speed, our oracle is defined as follows:

$$\forall t \in T, \forall p \in \mathcal{S}(r_t, s_t), \|h_t - p\| > 0 \quad (1)$$

with T observation period, $\|\cdot\|$ magnitude of the distance between locations in the (three-dimensional) collaborative space, r_t, h_t location at time t of the robot and the human agents, respectively, and s_t speed of the robot at time t. We let the reader refer to Di Cosmo et al. [16] for further details about the function $\mathcal{S}$ that calculates the safety zone.

The goal of testing process is to generate realistic scenarios that cause the system under test to violate the requirement in Eq. 1, thus affecting the safety of the human agent. More precisely, given a parametric scenario such as the collaborative task described in Sec. III, we define a *test case* as a value assignment:

$$A_i = \{f_1 = a_1, ..., f_n = a_n\} \quad (2)$$

where $f_1, ..., f_n$ is a set of domain-relevant and human-interpretable factors that control the scenario itself (e.g., light intensity, initial position of the human, speed of the robot, etc.). These factors are henceforth referred to as *domain features*. If the execution of the parametric scenario under the assignment A_i satisfies Eq. 1, the execution *succeed*s, it *fail*s otherwise. When the execution fails, we say that a safety hazard is detected.

In addition to the aforementioned testing goal, we also want to understand the extent to which generated test cases are diverse. Indeed, diversifying the regions where the domain features are sampled has advantages. This ensures a more effective exploration of alternative (i.e., diverse) circumstances leading to safety hazards. Diversity is especially beneficial to fault detection and debugging purposes. If the degree of

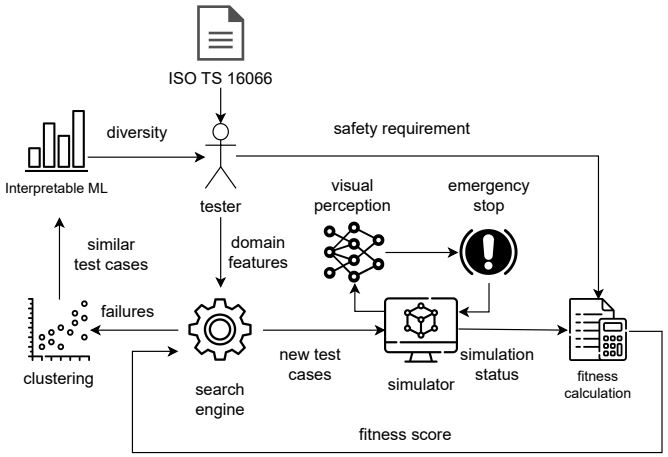


Fig. 3: Overview of our approach.

diversity is low, inputs are sampled from contiguous regions of the domain features and they are likely to trigger the same behavior. This means that most of the testing effort is wasted in generating very similar cases. More precisely, after generating and executing the test cases, we want to collect all the failure-revealing test cases $\{A_1, ..., A_n\}$ and study the degree of diversity of the dataset to select testing strategies that promote higher diversity.

B. Risk-driven Online Testing

Here we describe our current solution to the problem of failure-revealing test case generation by discussing our testing process and the specific search algorithms adopted by it.

We address the problem of failure-revealing test case generation using single-objective metaheuristic optimizing search algorithms. Our approach pushes the metaheuristic search toward test cases that are likely to increase the risk according to the SRMS operating mode, thus increasing the likelihood of breaking the requirement in Eq. 1.

Figure 3 shows an overview of our approach. The relevant parts for test case generation and execution are the search engine, the simulator, the ML visual perception component and emergency stop mechanism under test, and finally the fitness calculation. The process starts with the tester defining the domain features, the oracle, and the fitness function. Then, the search engine generates a set of new test cases $\{A_1, ..., A_n\}$ each one of them representing a different assignment to domain features. Each test case is used as input to run the parametric scenario in the simulator. The simulator feeds the ML component with a stream of images generated by the vision sensor. The ML component may trigger the emergency stop that instructs in turn the simulated physical elements of the robot. The simulation is monitored to check the oracle condition in Eq. 1 and then determine the outcome. The fitness is calculated from the status ω of the simulation by using the following function:

$$f(\omega) = \min_{t \in T, p \in \mathcal{S}(\omega.r_t)} \|\omega.h_t - p\| \quad (3)$$

TABLE II: Domain features in our case study.

domain feature	type	lower bound	upper bound
diffuse light (R)	float	0.0	1.0
diffuse light (G)	float	0.0	1.0
diffuse light (B)	float	0.0	1.0
human speed (m/s)	float	0.1	0.5
robot speed (m/s)	float	0.05	0.5
wait time human (s)	integer	1	50
wait time robot (s)	integer	1	50

TABLE III: Test case example in our case study.

domain feature	value
diffuse light 1	(0.1, 0.2, 0.3) RGB
human speed	0.33 m/s
robot speed	0.25 m/s
wait time human	2 s
wait time robot	5 s

where $\omega.r_t$ and $\omega.h_t$ represent the location of the robot and the human agents, respectively. The status ω records, for a given execution, the time series of the agents' location (i.e., the location of the agents for all t in the observation window T). Thus, the minimum distance observed during a simulation is used as fitness score (i.e., the closer the collaboration, the higher the risk). The fitness is fed back to the search engine that generates new test cases *minimizing* the score according to the adopted search algorithm. The optimization process terminates when the testing budget is finished. The budget is defined here in terms of generated test cases with an upper bound for the execution time.

Table II lists the domain features we defined in our case study. Each domain feature has its own type as well as its range of values defined by the tester according to his/her own domain knowledge. As an example, the parametric scenario has three *diffuse light* parameters defined by RGB float values (each one ranging in $[0, 1]$). As listed in Table II, we defined 7 domain features in total generating a nontrivial space mixing both discrete and dense domains. Table III shows an example of test case that can be executed by the simulator. In this example, the robot moves faster than the human, and the human starts the task before the robot.

Even though the SRMS operating mode imposes continuous satisfaction of the protective distance, there are some specific corner cases that we consider less dangerous, or alternatively, the system under test is not directly accountable for such hazards. In particular, according to our scenario description in Sec. III, the human agent is designed to ignore the behavior of the robot and possible hazardous moves. Therefore, the human may hit the robot even when the robot stands still after a successful emergency stop. Thus, we designed our oracle in a way it can easily detect these cases by taking into account the speed of the robot when a violation of Eq. 1 occurs. If the speed of the robot is equal to zero, it means the robot stands still and, therefore, the system is not accountable for the hazard. We classify this corner case as *false failure*.

C. Test Case Diversity Analysis

In the following, we discuss our test case diversity analysis approach including the main elements: diversity clustering and diversity validation.

1) Analysis Process: Studying the diversity is motivated by the practical need of investigating whether, in the context of ML-enabled systems, metaheuristic optimizing search can unveil diverse sets of failing scenarios or they steer the generation toward similar ones only. Indeed, testing approaches are often evaluated using traditional effectiveness mainly based on the number of detected failures [18]. However, identifying the best testing algorithm based on such effectiveness metrics may be misleading since the search process may generate lot of inputs sampled from contiguous regions of the search space triggering (almost) the same behavior. In such a case, substantial testing effort may be used to generate redundant cases (i.e., already seen during the search process).

For this reason, the adoption of search algorithms able to balance (1) quick convergence towards global optima, and (2) further search towards unexplored regions is crucial to ensure high effectiveness and high degree of diversity.

Figure 3 shows that our approach collects all failure-revealing test cases and then analyzes them by using clustering and intepretable ML. This latter step validates whether different clusters capture sets of test cases that are different in a relevant way, that is, different failure explanations. Indeed, similarity metrics that are purely based on the distance between test cases may hide important information [19]. Considerable distance does not always ensure relevant difference, while small distance may yield very diverse behavior due to non-linear, non-convex behavior of the ML components [18]. For this reason, we deliberately avoid favoring input diversity in the search process itself since this may lead to discarding similar inputs that yield diverse failures.

2) Diversity Clustering: According to Fig. 3, we run clustering to group together similar test cases. We rely on DBSCAN, a density-based clustering method that uses a parameterized distance metric to define clusters that are dense, in the sense that they have a high concentration of data points in a particular regions, separated by low-density regions. The selection of DBSCAN is motivated by its capability of (1) automatically determine the number of clusters, (2) identifying clusters of arbitrary shapes and sizes. In our context, the dataset of failure-revealing test cases contains value assignments to domain features (e.g., see Table III) that may be high dimensional. In this case, we can limit the computational complexity by reducing the number of dimensions before clustering with PCA. Principal components are ranked in the order of the amount of variation they capture, with the first component capturing the most variation, the second capturing the next most, and so on. Our approach adopts a common practice and retains only the first few principal components capturing 95% of the variance [14].

At the end of the clustering analysis, we use the total number of clusters as surrogate measure of test case diversity.

3) Diversity Validation: To measure the extent to which different clusters capture diverse sets of test cases, we make use of LIME explanations as mentioned in Sec. II. The idea is to build a local explanation for the outcome of each test case. The explanation includes weights to highlight the important features as well as their influence on the outcome (the higher weight the higher the influence). As an example, consider two test cases A_1 and A_2 belonging to different clusters. The local explanation for A_1 may reveal that when the robot speed is very high the feature has high influence, no matter what is the light condition. The explanation for A_2 may reveal instead that low light condition has the highest influence, under low-to-medium robot speed. In this case, the diversity between A_1 and A_2 holds since they belong to different clusters and the observed failure has different root cause(s) according to LIME.

Based on such reasoning, we introduce the notion of Local Explanation Diversity (LED) that we use to prove the validity of using the number of clusters as a surrogate measure of test case diversity in our context.

Given two test cases, we measure the difference between the corresponding LIME explanations by using the normalized *Levenshtein* distance [20] of the two sorted sequences of features obtained by ranking them according to the weights given by LIME. Thus, given two sets of test cases C and C', the LED measure is defined as the average pairwise distance between the test cases belonging to C and C', respectively:

$$LED = avg(\{L^*(seq(A), seq(A')) \; \forall A \in C, A' \in C'\}) \quad (4)$$

where seq represents the sorted sequence of feature for a test case, and L^* is the normalized *Levenshtein* distance. The higher the LED the higher the diversity between two sets.

Notice that, in our context, we want to measure the diversity of clusters of test cases. In particular, we talk about inter-cluster diversity if C and C' are different clusters. We talk about intra-cluster diversity if C and C' identify the same cluster.

V. Empirical Evaluation

In this section, we report on the empirical evaluation of our approach applied to the case study presented in Sec. III.

A. Research Questions

Our empirical evaluation has been designed to investigate the following Research Questions (RQs).

RQ1: *What is the effectiveness of the risk-driven test case generation across different search strategies?*

We compare multiple search strategies using a traditional effectiveness measure based on number of hazards (i.e., safety requirements violations) identified during a test session. We also assess the statistical significance and effect size of our results.

RQ2: *Are generated test cases diverse in a relevant way according to explanations of hazards?*

We study the results obtained by applying diversity clustering of the test cases and then diversity validation using LIME explanations and the LED measure.

TABLE IV: Configuration parameters of the search algorithms.

Algorithm	Configuration parameters
GA	Polynomial mutation probability 0.143 and DI 100.0, Binary crossover probability 0.9 and DI 100.0
ES	Polynomial mutation probability 0.143, Elitist option *true*, $\lambda = 20$, $\mu = 20$
SA	Temperature $T_0 = 1.0$, Min temperature 0.000001, Temperature variation coefficient $\alpha = 0.95$, Polynomial mutation probability 0.143

B. Design of the evaluation

We designed and carried out an experimental campaign[4] to evaluate our approach to risk-driven test case generation and diversity analysis. We executed multiple runs of the generation process by using different search approaches. In particular, we use and compare four selected single-objective search algorithms: three metaheuristic optimizing search algorithms including Genetic Algorithm (GA), Evolutionary Strategy (ES), Simulated Annealing (SA), and finally a (uniform) Random Search (RS) algorithm that we use as a baseline in our empirical evaluation. To implement these algorithms we rely on an existing off-the-shelf framework JMETALPY[5]. We configured the algorithms by following the best practices suggested in the documentation of the library. Table IV lists the main configuration parameters we used in our search engine during the experimental campaign. Concerning GA, we used an initial and offspring population size of 20 individuals and a total of 20 iterations (i.e., 400 individuals in total). Concerning the other algorithms (RS, ES, SA), we set population size and/or number of iterations accordingly, to be consistent with the total number of generated test cases. For the other parameters (e.g., mutation and crossover in GA, or initial temperature in SA), we used the values recommended in the original studies as stated by the documentation of JMETALPY (see Table. IV). For all search algorithms, a preliminary evaluation in which we executed a few proof-of-concept runs showed that 12 hours are enough for the runs to converge.

We henceforth refer to *run* as a whole generation and execution process adopting a given search algorithm, that is, the generation and execution of 400 test cases. To account for randomness, we repeated each run 20 times (i.e., $20 \times 400 = 8000$ test cases per individual search algorithm). Collected data out of all runs have been analyzed using the non-parametric Mann-Whitney U-test [21] to assess the statistical significance of the results between search strategies (significance level $\alpha = 0.05$). We use Vargha and Delaney's $\hat{A}_{AB}$ measure [22] to capture the effect size of the difference between A and B. We adopt the following standard classification: effect size $\hat{A}_{AB}$ ($= 1 - \hat{A}_{BA}$) is small, medium, and large when its value is greater than or equal to 0.56, 0.64, and 0.71, respectively.

[4]Replication package available at https://zenodo.org/record/8152294.
[5]https://github.com/jMetal/jMetalPy

TABLE V: Statistical analysis results.

Groups		Mann-Whitney U-test		$\hat{A}_{AB}$ effect size	
A	**B**	U statistic	p-value	estimate	magnitude
GA	RS	353.0	0.000	0.88	L
ES	RS	400.0	0.000	1.0	L
SA	RS	374.0	0.000	0.94	L
GA	ES	168.0	0.394	0.58	S
GA	SA	142.0	0.119	0.64	M
ES	SA	251.5	0.168	0.63	S

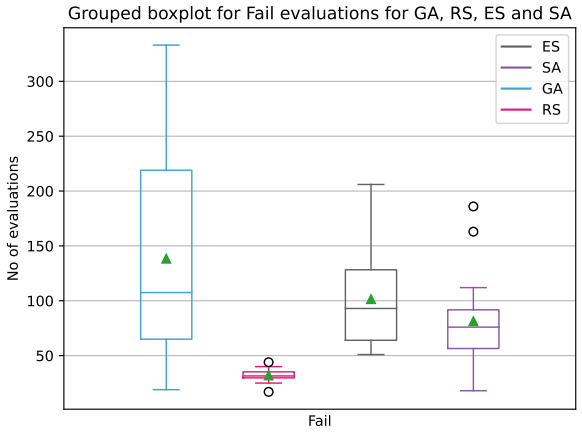
All the runs have been executed on a commodity hardware machine equipped with an Intel Core i7-8650U vPro and 24GB RAM, running UBUNTU 22.04.1 LTS. The simulation of our case study has been implemented using the robot simulator COPPELIASIM [9] (formerly known as V-REP) Edu v4.2 connected to an external test case generation PYTHON v3.8 program relying on the JMETALPY framework v1.5.7 and communicating with the simulator via ZEROMQ API [23].

C. Results

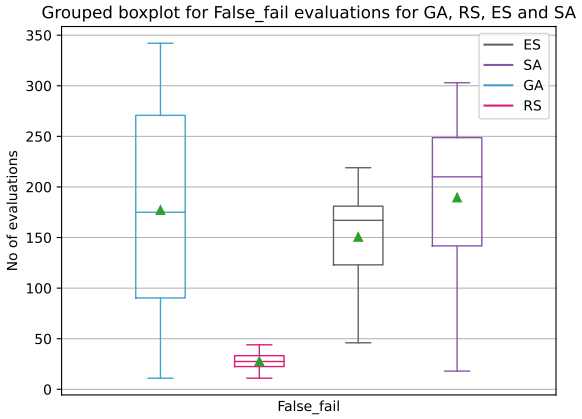
1) Effectiveness (RQ1): Figure 4 shows the distribution of the search algorithms per each individual test case outcome (either pass, fail, or false fail) using box plots. Results have been collected considering all the 20 runs for each search algorithm. The distribution in Fig. 4a shows that the effectiveness of RS is generally lower than the metaheuristic optimizing search algorithms with a median of detected hazards equal to 31.5, while it is equal to 76.0, 93.0, and 107.5 for SA, ES, and GA respectively. Concerning the three latter search algorithms, we can also observe that the dispersion of the distribution is the highest using GA and lowest using ES. GA yields the highest effectiveness on average since it exhibits the highest dispersion but also higher peaks.

Considering the results in Fig. 4c, we can observe that RS consistently generates more safe executions with a median value equal to 341.0. GA is instead the one associated with the lowest *pass* median value equal to 84.4. Again, SA is the search algorithm with the highest dispersion according to Fig. 4c. Concerning the *false fail* corner case, we can observe from Fig. 4b that SA yields the highest median equal to 210.0, while RS consistently generated fewer false hazards.

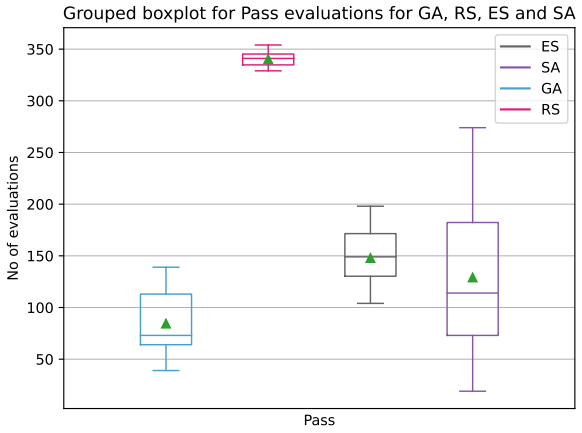
To assess the statistical significance of the effectiveness results, we followed the guideline presented by Arcuri and Briand [24] to perform a pairwise comparison between randomized search algorithms employed by the risk-driven test case generation process. As anticipated in Sec. V-B, we use the non-parametric Mann-Whitney U test since we do not assume that effectiveness data are drawn from a given parametric family of probability distributions. Thus, we compared the number of failures collected for all runs, for each pair of search algorithms A and B, as reported in Table V. We set a confidence level of 95% (i.e., p-value less than or equal to 0.05). Additionally, we determine the effect size using the Vargha-Delaney $\hat{A}_{AB}$ as a measure of the practical significance between the two populations A and B.



(a) FAIL Evaluations



(b) FALSE FAIL Evaluations



(c) PASS Evaluations

Fig. 4: Distribution of the search algorithms per individual test case outcome (fail, false fail, pass).

Table V shows that the difference between the baseline RS and all the other search algorithms is always statistically significant with a large (L) effect size. The comparison of GA, ES, and SA against each other does not achieve statistical significance (p-value between 0.119 and 0.394). Furthermore, the effect size is small (S), medium (M) and small (S): 0.58, 0.64, 0.63 when comparing GA and ES, GA and SA, and ES and SA respectively.

RQ1 Summary. The difference between the baseline RS and all the other search algorithms is statistically significant with large effect size. GA yields the highest average even though the pairwise comparison considering ES and SA is not significant and has a small effect size. The comparison between ES and SA is again not statistically significant and yields a small effect size.

2) Test case diversity (RQ2): To answer this RQ, we collected all the results obtained considering all runs, for all search algorithms. Thus, for each algorithm, we built a dataset containing all the test cases (i.e., value assignment to domain features) associated with outcome *fail*, according to Eq. 1. As anticipated in Sec. IV, the rationale for this process is the collection of all hazards observed during the test sessions (adopting a given search strategy). Each test case is a specific assignment to domain features and, therefore, the analysis of these assignments allows us to assess the similarity of generated test cases. As anticipated in Sec. IV, clustering automatically groups sets of similar data points. Thus, we use the total number of clusters as a surrogate measure of the test case diversity assuming that different clusters effectively capture diverse failures.

To validate this assumption, we measured the LED after collecting the results of clustering. Figure 5 shows the outcome of the diversity validation illustrating the distribution of both intra- and inter-cluster LED values. We can observe that results are consistent across alternative search algorithms. The median intra-cluster LED is around 0.25, while the median inter-cluster LED is around 0.65. This means that for all search algorithms, test cases drawn from different clusters are likely to be different in a relevant way, since, according to local explanations, they yield failures for different root causes (LED up to 0.65). Test cases drawn from the same clusters have in general higher similarity. Yet, there are still test cases in the same cluster that yield considerable differences (LED up to 0.25). This means that discarding similar test cases during the test case generation process may lead to overlooking significant differences when considering the root causes of failures.

Figure 6 shows the distribution of the number of clusters obtained by using DBSCAN for each search algorithm. All search algorithms are better compared to the baseline RS. We can observe that ES generally exhibits a higher number of clusters compared to GA and SA. SA is also slightly better than GA. Since the intra- and inter-cluster diversity is consistent across these algorithms, we can conclude that even though GA yields more failures on average (see Fig. 4a), both ES and SA are better when it comes to failure diversity. In particular, ES is the best algorithm according to our results. This reinforces the fact that selecting the testing strategy based on the number of failures only may be misleading as a lot of failures does not necessarily mean comprehensive coverage of

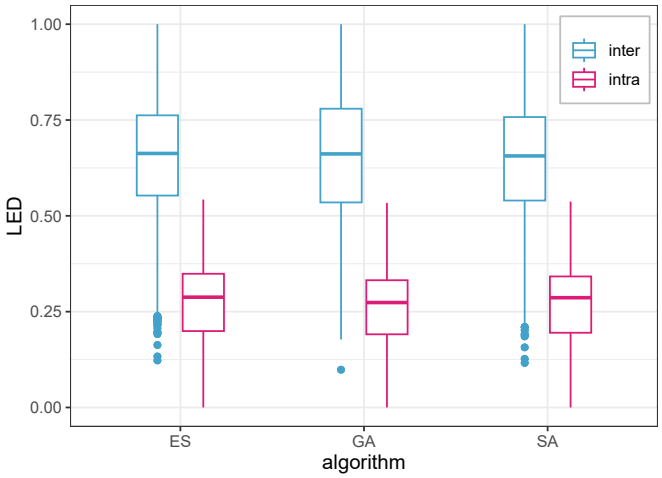


Fig. 5: Intra- and inter-cluster diversity validation.

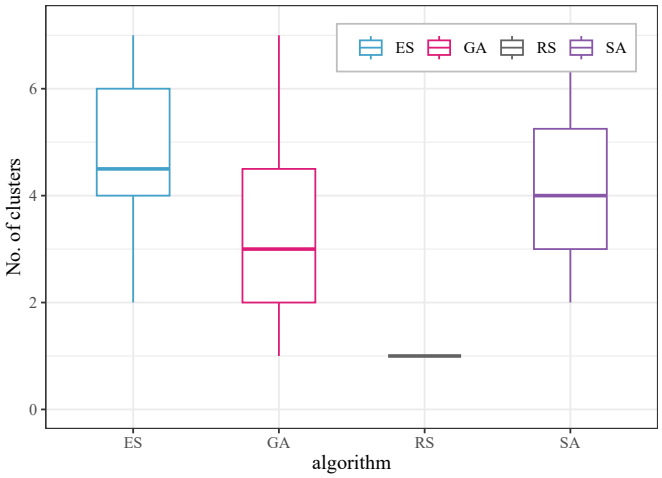


Fig. 6: Number of clusters per run.

a wide range of relevant failures.

Table VI shows a summary of the results obtained by DBSCAN clustering with PCA dimension reduction with explained variance ratio at a confidence level of 95% when considering the testing outcomes collected from all the runs ranked according to the total number of clusters per algorithm. Results are consistent to those obtained without dimension reduction. We can observe that ES yields more clusters (82) than the other algorithms. ES is followed by SA (67 clusters), and then GA (54 clusters). The baseline RS is the last one in the rank and represents the worst case with only 1 cluster.

TABLE VI: Clustering results with dimension reduction

Algorithm	#Clusters w PCA	#PCA components
ES	82	2
SA	67	2
GA	54	2
RS	1	2

RQ2 Summary. The LED measure is consistent across all selected search algorithms. Test cases drawn from the same clusters have in general higher similarity. Yet, there are still test cases in the same cluster that yield considerable differences. Thus, discarding similar test cases may lead to overlooking significant differences in the testing outcome. Even though GA yields in general more failures, both ES and SA lead to more clusters (more diversity). Thus, selecting the testing strategy based on the number of failures only may be misleading.

D. Threats to validity

In the following, we discuss major threats to the validity of our empirical evaluation in the common categories: internal, external, construct, and conclusion.

a) External Validity: Threats in this category may emerge if the characteristics of our case study cannot be generalized to other systems. Since we use an individual CAIS and a specific robot simulator (i.e., CoppeliaSim), external validity is our main concern. To the best of our knowledge, the case study adopted in this work is a representative example of CAIS in the Industry 4.0 [25] and by extension, Industry 5.0 [26], [27] including a nontrivial (dense) search space with 7 relevant domain features. CoppeliaSim represents a high fidelity robot simulator used by leading companies to simulate industrial applications. Other publicly available high-fidelity robot simulators exist. For instance, Gazebo[6] could be adopted to carry out further experiments on this topic. Note that such experiments are highly computationally-intensive. Our experimental campaign took around 45 computing hours in total (around 12 for GA and ES, 11 for SA, 10 for RS). Nevertheless, additional experiments with different case studies and high-fidelity simulators would be required to increase the generalizability of our findings.

b) Internal Validity: Threats may be caused by bias in establishing cause-effect relationships in our experiments. To limit these threats, we elicited and controlled the factors of interest as much as possible during our experimental campaign. In particular, the search algorithms have direct access to domain features to find cause-effect relations between generated test cases and the fitness measured during the simulations. We also directly controlled the specific search algorithms adopted during the experiments by using the same configuration parameters across all the runs (see Table. IV). Direct manipulation of these factors increases internal validity compared to observations without manipulation.

c) Conclusion Validity: We addressed major threats in this category by reducing the possibility of obtaining results by chance. In particular, for each run we collected a large sample of test cases (400). Furthermore, each run has been repeated 20 times for all algorithms, executing a total amount of $32k$ test cases (i.e., 20 runs, 4 search algorithms, and 400 test cases generated for each run). We followed well-known guidelines in the software engineering research community to

[6] https://gazebosim.org/

assess the statistical significance of our experiments [24]. In particular, we conducted pairwise comparisons using the non-parametric Mann-Whitney U test to calculate the p-value. In addition to statistical significance, we used the Vargha and Delaney's $\hat{A}_{AB}$ non-parametric effect size measure.

d) Construct Validity: We limited this threat by assessing the validity of our selected metrics before using them in our experiments. In particular, we measured the effectiveness of test sessions by counting the number of failures (i.e., safety violations in our case) that represents a standard metric. According to the guidelines in [21], we used the standard significance level $\alpha = 0.05$ for the Mann-Whitney U test. We also followed the common categories (small, medium, large) and corresponding levels (0.56, 0.64, 0.71) for the Vargha and Delaney's $\hat{A}_{AB}$ non-parametric effect size measure as described in [22]. To measure test case diversity we adopted the number of clusters. It is worth noting that this is a non-standard surrogate measure of diversity that has been validated by studying intra- and inter-cluster LED.

VI. Related Work

Online testing of ML-enabled systems in a closed loop with a simulated environment is an active research area, especially in the automotive domain. Abdessalem et al. [28] studied the problem of testing control systems of autonomous vehicles leveraging vision-based ML components. The approach combines evolutionary search and classification models (Decision Tree) to characterize critical regions used by the evolutionary search itself. The authors show that their approach generates more distinct test scenarios compared to baseline approaches. Gambi at al. [19] propose the combination of content generation and search-based testing to create virtual roads for autonomous vehicles. Their approach embeds a similarity measure to filter out similar test cases during the search process. AV-FUZZER [29] is an online testing framework that applies fuzzing techniques to modify nominal traffic conditions and create situations in which an autonomous vehicle can run into safety violations. Evaluation of the testing algorithms is based on the number of failures. Haq et al. [18] proposed to combine surrogate-assisted and many-objective optimization to efficiently spot safety requirement violations in autonomous driving scenarios. The authors compare different testing strategies using the proportion of safety requirements that are violated when running the test suite over the total number of safety requirements.

Overall, we found that failure diversity in ML-enabled systems is less studied compared to the ability to produce a lot of failures. However, this concern is recently emerging. A notable example in this direction is DeepHyperion [6], a testing framework for ML-enabled systems that determines failure diversity by calculating the ratio of the maximum Manhattan distance between cells in the feature map containing failures and the total number of mapped misbehaviors.

Considering our target domain of interest, collaborative robotics, there is a growing concern for human-centric engineering approaches. In this direction, the detection of safety issues has been recently studied by Huck et al. [30], [31]. The authors use a risk metric to guide the search of possible human behaviors toward high-risk behaviors which are more likely to expose hazards. Compared to our work, the main focus of testing is on human behaviors only, neglecting other factors of the environment. Parisi et al. [32] consider the problem of online prediction of failures in collaborative robots that carry out force-sensitive tasks. Failure prediction uses unsupervised ML, with learning parameters optimized via GA. In these lines of research, robots have pre-programmed behaviors and do not embed ML components. Furthermore, the diversity of detected failures is not taken into account.

VII. Conclusion

We introduced a novel risk-driven online testing method and test case diversity analysis for ML-enabled systems running in safety-critical settings. Due to the safety-critical nature of these systems, we rely on a high-fidelity (industry-strength) simulator and we apply an online search-based testing process that aims at falsifying domain-specific safety requirements. We also studied the diversity of failure-revealing test cases generated by different search algorithms, including genetic algorithm, simulated annealing, evolutionary search, and random search (our baseline). To study the degree of diversity our method combines clustering and model-agnostic interpretable ML to group together test cases with a small distance and then validate whether such test cases are similar in a relevant way, that is, they yield the same hazard for the same root causes.

We applied our approach to a CAIS case study featuring a collaborative robot application, where an ML visual perception component is used to detect foreign objects and trigger emergency stop actions of the robot. We conducted an empirical evaluation with the aim of studying: (1) the effectiveness of the risk-driven test case generation across different search strategies and the statistical significance of the results; (2) the diversity of the generated test cases causing hazards. Metaheuristic optimization search approaches are better compared to the baseline with statistically significant results. We have also demonstrated that selecting a particular search strategy based on traditional metrics of effectiveness only may be misleading. Furthermore, discarding similar test cases may lead to overlooking significant differences in the testing outcome due to non-linear, non-convex behavior of ML components.

We plan to expand the scope of our experiments by making the scenario more complex and considering different application domains. We also plan to develop novel methods that favor test case diversity by design by embedding local explanations within the test case generation process.

Acknowledgments

This work has been partially supported by the Austrian Science Fund (FWF) for the project SafeSec, grant agreement No: I 4701 Internationale Projekt.

References

[1] J. G. Adigun, M. Camilli, M. Felderer, A. Giusti, D. T. Matt, A. Perini, B. Russo, and A. Susi, "Collaborative artificial intelligence needs stronger assurances driven by risks," *Computer*, vol. 55, no. 3, pp. 52–63, 2022.

[2] M. Camilli, M. Felderer, A. Giusti, D. T. Matt, A. Perini, B. Russo, and A. Susi, "Risk-driven compliance assurance for collaborative ai systems: A vision paper," in *Requirements Engineering: Foundation for Software Quality* (F. Dalpiaz and P. Spoletini, eds.), (Cham), pp. 123–130, Springer International Publishing, 2021.

[3] F. U. Haq, D. Shin, S. Nejati, and L. Briand, "Can offline testing of deep neural networks replace their online testing?," *Empirical Software Engineering*, vol. 26, p. 90, Jul 2021.

[4] M. Camilli, R. Mirandola, and P. Scandurra, "Taming model uncertainty in self-adaptive systems using bayesian model averaging," in *Proceedings of the 17th Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '22, (New York, NY, USA), p. 25–35, Association for Computing Machinery, 2022.

[5] M. Camilli, R. Mirandola, and P. Scandurra, "Enforcing resilience in cyber-physical systems via equilibrium verification at runtime," *ACM Trans. Auton. Adapt. Syst.*, feb 2023. Just Accepted.

[6] T. Zohdinasab, V. Riccio, A. Gambi, and P. Tonella, "Deephyperion: Exploring the feature space of deep learning-based systems through illumination search," in *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2021, (New York, NY, USA), p. 79–90, Association for Computing Machinery, 2021.

[7] K. Khan, S. U. Rehman, K. Aziz, S. Fong, and S. Sarasvady, "DBSCAN: Past, present and future," in *The Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014)*, pp. 232–238, 2014.

[8] C. Molnar, *Interpretable Machine Learning*. 2 ed., 2022.

[9] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1321–1326, IEEE, 2013.

[10] P. McMinn, "Search-based software testing: Past, present and future," in *2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops*, pp. 153–163, 2011.

[11] C. C. S. Liem and A. Panichella, "Oracle issues in machine learning and where to find them," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, ICSEW'20, (New York, NY, USA), p. 483–488, Association for Computing Machinery, 2020.

[12] M. Z. Rodriguez, C. H. Comin, D. Casanova, O. M. Bruno, D. R. Amancio, L. d. F. Costa, and F. A. Rodrigues, "Clustering algorithms: A comparative approach," *PloS one*, vol. 14, no. 1, p. e0210236, 2019.

[13] S. Velliangiri, S. Alagumuthukrishnan, and S. I. Thankumar joseph, "A review of dimensionality reduction techniques for efficient computation," *Procedia Computer Science*, vol. 165, pp. 104–111, 2019. 2nd International Conference on Recent Trends in Advanced Computing ICRTAC -DISRUP - TIV INNOVATION , 2019 November 11-12, 2019.

[14] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Philos Trans A Math Phys Eng Sci*, vol. 374, p. 20150202, Apr. 2016.

[15] M. Camilli, M. Felderer, A. Giusti, D. T. Matt, A. Perini, B. Russo, and A. Susi, "Towards risk modeling for collaborative ai," in *2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN)*, pp. 51–54, IEEE Computer Society, 2021.

[16] V. Di Cosmo, A. Giusti, R. Vidoni, M. Riedl, and D. T. Matt, "Collaborative robotics safety control application using dynamic safety zones based on the iso/ts 15066: 2016," in *Advances in Service and Industrial Robotics: Proceedings of the 28th International Conference on Robotics in Alpe-Adria-Danube Region (RAAD 2019) 28*, pp. 430–437, Springer, 2020.

[17] K. Pulli, A. Baksheev, K. Kornyakov, and V. Eruhimov, "Real-time computer vision with opencv," *Commun. ACM*, vol. 55, p. 61–69, jun 2012.

[18] 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*, ICSE '22, (New York, NY, USA), p. 811–822, Association for Computing Machinery, 2022.

[19] A. Gambi, M. Mueller, and G. Fraser, "Automatically testing self-driving cars with search-based procedural content generation," in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2019, (New York, NY, USA), p. 318–328, Association for Computing Machinery, 2019.

[20] L. Yujian and L. Bo, "A normalized levenshtein distance metric," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1091–1095, 2007.

[21] R. R. Wilcox, *Fundamentals of modern statistical methods: Substantially improving power and accuracy*, vol. 249. Springer, 2001.

[22] R. Grissom and J. Kim, *Effect Sizes for Research: A Broad Practical Approach*. Lawrence Erlbaum Associates, 2005.

[23] C. Robotics, "Zeromq remote api." Accessed: 20 January 2023.

[24] A. Arcuri and L. Briand, "A practical guide for using statistical tests to assess randomized algorithms in software engineering," in *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, (New York, NY, USA), p. 1–10, Association for Computing Machinery, 2011.

[25] S. Vaidya, P. Ambad, and S. Bhosle, "Industry 4.0 – a glimpse," *Procedia Manufacturing*, vol. 20, pp. 233–238, 2018. 2nd International Conference on Materials, Manufacturing and Design Engineering (iCMMD2017), 11-12 December 2017, MIT Aurangabad, Maharashtra, INDIA.

[26] E. Commission, D.-G. for Research, Innovation, and J. Müller, *Enabling Technologies for Industry 5.0 : results of a workshop with Europe's technology leaders*. Publications Office, 2020.

[27] E. Commission, D.-G. for Research, Innovation, M. Breque, L. De Nul, and A. Petridis, *Industry 5.0 : towards a sustainable, human-centric and resilient European industry*. Publications Office, 2021.

[28] R. Ben Abdessalem, S. Nejati, L. C. Briand, and T. Stifter, "Testing vision-based control systems using learnable evolutionary algorithms," in *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, pp. 1016–1026, 2018.

[29] G. Li, Y. Li, S. Jha, T. Tsai, M. Sullivan, S. K. S. Hari, Z. Kalbarczyk, and R. Iyer, "Av-fuzzer: Finding safety violations in autonomous driving systems," in *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, pp. 25–36, 2020.

[30] T. P. Huck, C. Ledermann, and T. Kröger, "Testing robot system safety by creating hazardous human worker behavior in simulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 770–777, 2022.

[31] T. P. Huck, N. Münch, L. Hornung, C. Ledermann, and C. Wurll, "Risk assessment tools for industrial human-robot collaboration: Novel approaches and practical needs," *Safety Science*, vol. 141, p. 105288, 2021.

[32] L. Parisi and N. RaviChandran, "Genetic algorithms and unsupervised machine learning for predicting robotic manipulation failures for force-sensitive tasks," in *2018 4th International Conference on Control, Automation and Robotics (ICCAR)*, pp. 22–25, 2018.