SiMOOD: Evolutionary Testing Simulation with Out-Of-Distribution Images

Raul Sena Ferreira, Joris Guérin, Jérémie Guiochet, Hélène Waeselynck

SiMOOD: Evolutionary Testing Simulation with Out-Of-Distribution Images

Raul Sena Ferreira
LAAS-CNRS, University of Toulouse, Toulouse, France
rsenaferre@laas.fr

Joris Guerin
LAAS-CNRS, University of Toulouse, Toulouse, France
jorisguerin.research@gmail.com

Jeremie Guiochet
LAAS-CNRS, University of Toulouse, Toulouse, France
jeremie.guiochet@laas.fr

Helene Waeselynck
LAAS-CNRS, University of Toulouse, Toulouse, France
helene.waeselynck@laas.fr

***Abstract*—Testing perception functions for safety-critical autonomous systems is a crucial task. The reason is that accurate ML models applied in computer vision tasks still fail in scenarios where humans perform well. Out-of-distribution (OOD) images are usually a source of such failures. For this reason, literature usually applies data augmentation techniques or runtime monitors such as OOD detectors to increase robustness. Evaluating such solutions is usually performed by analyzing metrics based on positive and negative rates over a dataset containing several perturbations. However, using such metrics on such datasets can be misleading since not all OOD data lead to failures in the perception system. Hence, testing a perception system cannot be reduced to measuring Machine Learning (ML) performances on a dataset but rely on the images captured by the system at runtime. However, the amount of time spent to generate diverse test cases during a simulation of perception components can grow quickly since it is a combinatorial optimization problem. Aiming to provide a solution for this challenging task, we present SiMOOD, an evolutionary simulation testing of safety-critical perception systems, which comes integrated into the CARLA simulator. Unlike related works that simulate scenarios that raise failures for control or specific perception problems such as adversarial and novelty, we provide an approach that finds the most relevant OOD perturbations that can lead to hazards in safety-critical perception systems. Moreover, our approach can decrease, at least 10 times, the amount of time to find a set of hazards in safety-critical scenarios such as autonomous emergency braking system simulation. Besides, code is publicly available for use.**

***Index Terms*—Testing, Autonomous Vehicle, Out-Of-Distribution, Perception System, Computer Vision, Safe AI.**

I. Introduction

State-of-the-art Machine Learning (ML) models built with deep learning architectures perform reasonably well in safety-critical tasks such as perception in autonomous systems. However, these models tend to have problems such as ghost detections [1], misclassifications [2], or even being blind to the presence of new objects [3]. These problems usually come when the model is exposed to out-of-distribution (OOD) data, such as distributional shifts, sensor noise, and new classes [4], leading to wrong predictions even with high confidence associated with the decision [5].

To increase ML robustness, practitioners usually apply data augmentation techniques during training or OOD detectors during operation [3], [2]. The performance of such methods is usually demonstrated by inspecting the number of false positives and false negatives when exposed to several datasets containing common image perturbations and new objects [4], [6], [7]. However, simulation is an essential part of testing since not all ML errors lead to hazards, and most of the time, the physics of such autonomous systems need to be taken into account when searching for hazards. Therefore, in this work, we present SiMOOD, an approach dedicated to test perception functions built with ML. SiMOOD is integrated within CARLA simulator [8].

As illustrated in Figure 1, we apply OOD perturbations during simulation instead of using it over static datasets as it is usually done in the literature [9], [4], [10], [7]. However, since each OOD perturbation has several parameters, the number of possible simulations to find a hazard grows exponentially. Moreover, the time to simulate even a tiny part of these simulations grows quickly. To mitigate this problem, SiMOOD uses a two-step approach. First, it performs a unit testing of the ML model using a genetic algorithm (GA) [11] on the same dataset applied to train the ML model. It determines which combination of OOD perturbations with their respective intensity levels increases the number of incorrect predictions. Second, the selected perturbations are applied to the simulation to verify if they lead to hazards in the system. That is, SiMOOD takes a set of existing scenarios and injects image perturbations selected by a GA-on-data approach and posteriorly tests them on simulations, proving itself capable of turning safe scenarios into unsafe ones, which can be analyzed in order to improve the safety of the system.

We consider four categories of OOD perturbations: novel

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 812.788 (MSCA-ETN SAS). This publication reflects only the authors' view, exempting the European Union from any liability. Project website: http://etn-sas.eu/.

This research has also benefited from the AI Interdisciplinary Institute ANITI. ANITI is funded by the French "Investing for the Future – PIA3" program under the Grant agreement No ANR-19-PI3A-0004.

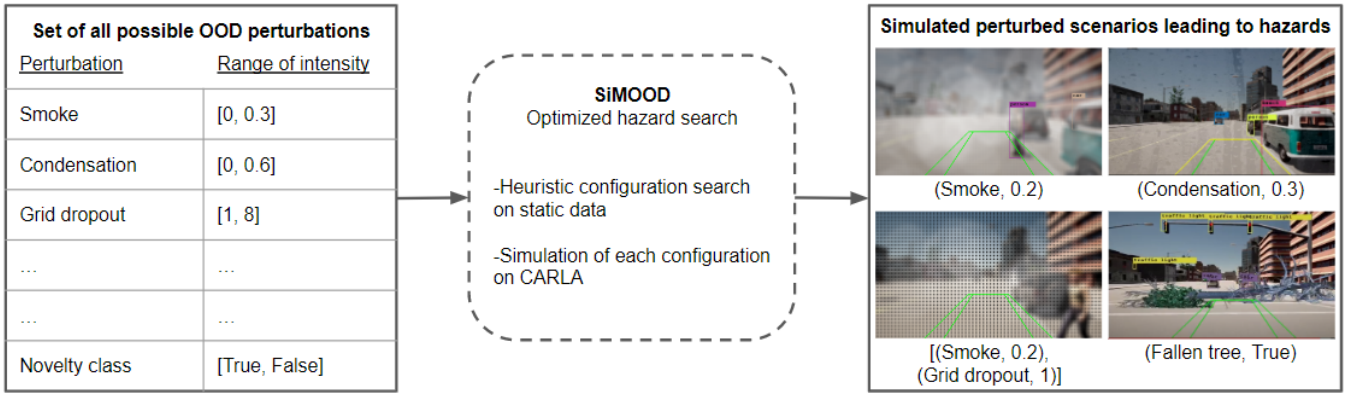


Fig. 1: SiMOOD simulation flow.

classes, distributional shifts, noise, and anomalies.

Our main contributions are:

- *Open-source testing approach integrated with CARLA simulator*: SiMOOD apply several relevant OOD perturbations at simulation time. Such perturbations also vary in intensity. Performing such tests yields more realistic results than performing them over static datasets. Besides, SiMOOD is open-source [12].
- *Optimized search for hazards*: SiMOOD is an off-the-shelf testing approach for perception functions in simulated environment. It uses an evolutionary approach to find the minimal OOD perturbations that may lead to hazards during simulation. Due to its GA-on-data approach, SiMOOD decreases 10 times the amount of time required to find a set of hazards in safety-critical tests while considering a huge search space.

This work is organized as follows: in Section II, we provide a brief explanation of the concepts used in our approach, the simulated OOD data, and existing related works. In Section III, we detail the architecture of our approach and introduce its main functionalities. In Section IV, we show the experiments and the insights provided by the results. Finally, in Section V, we present our final considerations and limitations.

II. Background

We focus on simulation testing of ML-based object detectors when exposed to out-of-distribution images.

A. Perception systems with ML-based object detectors

Supervised ML models for vision are usually divided into shallow and deep models. Both shallow and deep learning models are built to receive an instance (e.g., image or other sensory values) and perform a prediction of one/multiple object(s) on it. It is done, initially, by converting an image to a matrix of numerical values, and classifying this matrix in a categorical value that represents a specific class previously seen during the training process. Such models are usually validated by analyzing the discrepancy between the predicted objects and a ground-truth in a validation dataset. If the prediction error in the validation process is low enough to be considered satisfactory, the model goes to production. Since deep learning models demonstrated, on average, superior performance in computer vision than shallow ones, such deep learning architectures were integrated into several perception pipelines of many safety-critical functions of autonomous systems such as collision avoidance and path planning. In this work, we refer to ML models like the ones built on top of deep learning architectures. Hence, we perform our experiments using one of the most popular ML-based object detectors: YOLO [13]. Next, we explain which OOD data can negatively affect the ML model performance.

B. Out-of-distribution images

ML models tend to be biased to the training data [5], resulting in a natural inability of a model to generalize 100% of the time since rare object characteristics and conditions tend to be underrepresented and different enough to threaten the ML model performance. Such data is known as out-of-distribution data (OOD). Despite the existence of different classifications, we name here four common types of OOD data that an ML model can be exposed during simulation:

- **Novelty**: it happens when new objects appears at runtime. For example, an ML model can be trained to identify dogs but fails to classify a cat since it was not present in the training data (novelty class) [14].
- **Anomaly**: it is also known as corner cases [1]. It happens when the ML has to interpret an unexpected interaction between known objects, such as an overturned vehicle on the road [15].
- **Distributional shift**: it occurs when the distribution of the incoming data is different from the training, while the class generation mechanism keeps unchanged [16]. It means that at runtime, the incoming data can have different characteristics from the training set while its semantic content remains the same. Such condition decreases the ML performance through time in non-stationary environments [17]. For example, images that have a compromised visibility due to environmental conditions (e.g., snow, smoke, condensed water on the lens) [4]

- **Noise**: it is provoked by common image corruptions resulting from different types of sensor failures (e.g., Gaussian noise, occlusions by black pixels) [7].

Next, we show related works that propose simulation-based testing of perception systems.

C. Related works

We focus on simulation-based testing of perception systems composed of ML algorithms, specifically object detectors exposed to out-of-distribution images. Therefore, works that focus on finding safety violations on control [18], system specifications [19], error space exploration [20], path planning [21], and new driving scenes generation [22] are outside of the scope of this work, and therefore, not considered in this section.

As the first example of related work, Pei et al. [23] proposes DeepXplore, an automated white-box testing of deep learning systems. The authors argue that DeepXplore can measure code coverage by measuring the neuron coverage of a deep learning algorithm when perturbing the images that are fed into these ML models. Following a similar idea of neuron coverage, Tian et al. propose DeepTest [24], an automated testing approach of deep-neural-network-driven autonomous cars. The authors can find several ML failures in a test set in which the ML model should perform a steering angle action correctly. Zhang et al. [25] push forward the idea of perturbing images for testing by proposing DeepRoad: a GAN-based metamorphic testing and input validation framework for autonomous driving systems. DeepRoad generates driving scenes with different weather conditions by using generative adversarial networks (GAN) along with the corresponding real-world weather scenes. DeepXplore, DeepTest, and DeepRoad test safety-critical perception functions against generated OOD images. However, these works perform tests on datasets, and such offline testing alone cannot guarantee that safety violations will not happen during online testing [26]. Therefore, our work also differs from the previous related works.

Regarding related works that also propose simulation tests, Dreossi et al. [27] propose to test the safety-critical system when exposed to novel classes. The authors apply a falsification method over images with different light conditions and distances to find regions of uncertainty. After that, they test it against a simulator. Rossolini et al. [28] test the robustness of semantic segmentation models by applying adversarial colored patches on the simulation and real images. The authors present extensive experiments to validate the proposed attack and defense approaches in real-world scenarios. Boloor et al. [29] also test and simulate physical adversarial examples that can affect the detection performance for object detection tasks. Boloor et al. [29] apply black lines on the streets to foul the object detector and force it to go in a different direction. Fahmy et al. [30] propose a tool to debug DNNs for safety analysis. The tool finds clusters of images with common characteristics that lead to errors by using the information propagated by DNN neurons during the prediction. Finally, Haq et al. [26] showed that for continuous values, DNN prediction errors not identified by testing DNNs with static datasets can yield many safety violations during online testing. However, authors also emphasizes that such premise cannot be applied when testing perception-based tasks. Our work goes one step further on this problem by focusing on minimal image perturbations that leads to hazards during simulation.

Our approach differs from Rossolini et al. [28] since our focus is on the object detection task. Our approach also differs from [29], [27], [30] since these related works are focused on an in-depth analysis of one type of OOD data (e.g., adversarial, novelty, and distributional shift respectively). In contrast, we consider four types of OOD data that are a source of threats for perception systems with ML-based object detectors. We also provide a list of fine-grained intensity perturbations selected over a higher search space. Next, we present SiMOOD.

III. SiMOOD overview

SiMOOD generates hazardous perturbations in a specific scenario in four types of OOD data: novelty, anomaly, distributional shift, and noise. For novelty class and anomaly perturbations SiMOOD works with binary intensity levels (true/false) and continuous values for distributional shift and noise. For example, for novelty class experiments, SiMOOD can inject *a new object* from the operation domain design in a random point of the scenario, and for anomaly experiments, SiMOOD can inject *a known object* in a way that is not expected to be found in the original dataset. As noted, there is no sense in applying intensity levels for novelty and anomaly perturbations but rather applying random locations for the novel/anomalous objects in the scene.

For perturbations that can vary with a range of continuous values, SiMOOD needs to generate values that can lead to hazards in the simulation. Since the search space for such values are large, SiMOOD applies an evolutionary search to find the values that have more chance of provoking ML failures that lead to hazards in the simulation.

As illustrated in Figure 2, SiMOOD is divided in three parts: generation, simulation, and evaluation. Thus, during the generation, SiMOOD applies an evolutionary search to find the n-most relevant OOD combinations. Each combination represents a vector containing m tuples in the format (perturbation, intensity). Each perturbation has a range of intensity values. For instance, a single combination of size m=*2* can be formed by [(Gaussian noise, 0.2), (Blur, 3)]. Thus, we introduce the following definitions:

- **Genes**: A gene is represented as an OOD perturbation with its respective intensity value (e.g., Gaussian noise, 0.2). Each OOD perturbation varies in intensity from *0* (no effect) to L (severe but still interpretable).
- **Individual**: an individual is composed of a number of genes. For instance, an individual of size m=*2* is represented by a vector of m tuples: [(Gaussian noise, 0.2), (Blur, 1)].
- **Population**: it is a set of individuals. An initial population of size n means a generation of n-individuals.
- **Genetic algorithm iterations**: since the number of combinations and their parameters is large, the space search

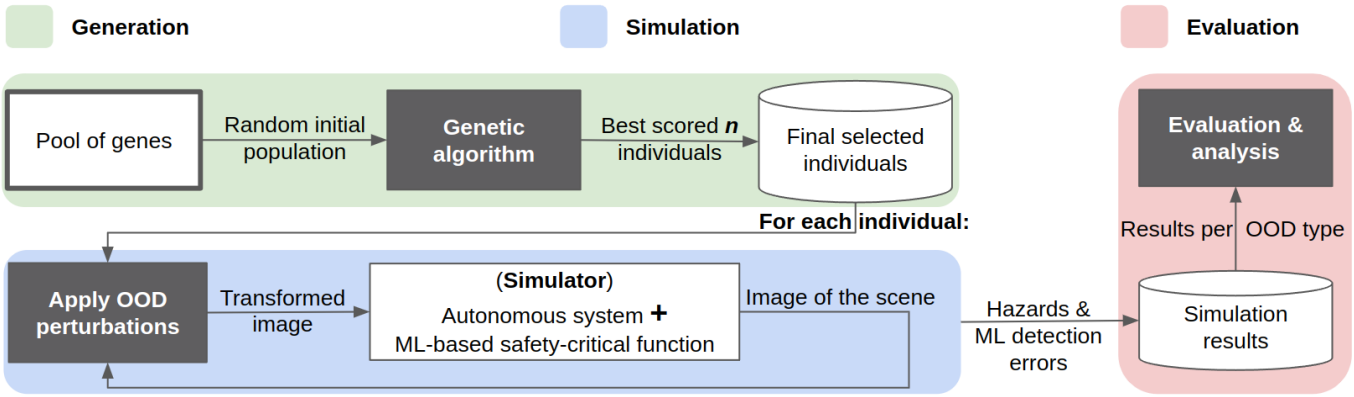


Fig. 2: SiMOOD overview: generation, simulation, and evaluation phases.

is similarly large. Therefore, in the first iteration, the GA randomly selects an initial population of size n.

After this generation step, the n-most relevant individuals selected by the GA are iteratively simulated in a safety-critical system. Each combination is applied to each frame. The transformed image is exposed to the ML-based safety-critical function, and the results are stored for future analysis. Next, we detail each part of SiMOOD.

A. Generation

During this part, SiMOOD performs the task of finding suitable configurations (e.g., combinations of OOD perturbations) using a genetic algorithm (GA) approach as illustrated in Figure 3. A GA comprises an initial population randomly selected from the original population. A population represents a group of individuals. Each individual comprises a group of genes (or a group of *parameters*). The GA *selects* the best individuals based on a *fitness score* outputted from a *fitness function*. After it, the k best selected individuals generate k-new individuals for the next generation. For each pair of selected individuals, a *crossover* point is chosen randomly from within the genes. Finally, a *mutation* can occur to maintain diversity within the population.

These perturbations are combinations of *distributional-shift* and *noise* perturbations. For example, Gaussian noise with a blur effect, or Spackle noise in an environment with heavy smoke. Since the ML always gives wrong predictions to new classes, the novelty class category is not applied in the GA but rather as an option when performing a simulation with our approach. Next, we translate the GA terms to our task as illustrated in Figure 3:

a) ***Initial population****:* it is randomly generated, and represents a subset of all combinations of OOD transformations.

b) ***Apply perturbations****:* it sequentially applies a combination of perturbations over an image following an order of occurrence of these perturbations inside of an individual. That is, an individual containing [(Blur, 0.1), (Smoke, 0.3)] leads to a blur transformation followed by a smoke transformation.

c) ***Fitness evaluation****:* each set of perturbations is applied to the same dataset used to train the ML model. Hence, we have n transformed datasets that are exposed to a fitness function. The fitness function uses the same ML model applied later in the simulation as part of the final fitness score. Thus, the fitness function calculates all true/false positives/negatives yielded by the ML model when exposed to these transformed datasets. Any classical ML metric value α between 0 and 1 can be used (e.g., mAP, false positive/negative rates (FPR) regarding the detection task [7], precision and recall).

The **fitness score** f is a sum of α and the normalized vector of perturbation intensities η multiplied by a smoothness term ω, that is, $f = \alpha + (\eta * \omega),\ \omega \in \mathbb{R}[0, 1]$. These terms are added to α to force the GA to reward perturbations that lead to hazards but with a minimum amount of intensity. Worth mentioning that if α is based on measuring the ML model errors (ex: mean average error) instead of the model correct predictions, this regularization term is decreased by α instead (e.g., $|(\eta * \omega) - \alpha|$).

To calculate η we first normalize the vector of perturbation intensities $v = v_1, ..., v_m$ through the formula $(val - min\_val)/(max\_val - min\_val)$, which val means the intensity value of a particular OOD perturbation, min_val and max_val represents using the minimum (e.g., 0, or no effect), and the maximum value of intensity for a particular OOD perturbation. Hence, after applying the above formula for each gene of an individual of size m, we obtain the final value by $\eta = \frac{1}{m} \sum_{i=1}^{m} v_i$.

Regarding ω, it is intended to penalize a high intensity η over the ML metric α. The value of ω can be increased if one wants to give more importance to η, or decreased otherwise. Besides, since our focus is to find different perturbations rather than find the highest intensity levels, we do not allow $\omega > 1$.

d) ***Selection****:* once all fitness scores are collected, the algorithm selects k best OOD perturbations that had the most relevant fitness scores. In this work, we choose to work with odd number of individuals (e.g., k=2) since it is a simple premise of the GA, which performs crossover between the best pairs of the generation. This criteria is also generic and

depends on the objective of the test. In our case, we want to find the minimal OOD perturbations that have a high probability of leading to hazards. Hence, the selection mechanism will choose one leading to the worst ML performances. For instance, if the fitness score is based on accuracy, SiMOOD selects k individuals that led to the worst results using minimal perturbation intensities.

e) ***Crossover****:* the selected individuals k generates k new individuals by exchanging their genes.

f) ***Mutation****:* it might occur on the resulting pair of individuals by changing a specific gene at an arbitrary point. Unlike the crossover, the mutation does not always happen and has a low probability of occurrence. The mutation randomly chooses a gene to replace and assigns a new random value to it. In our approach, to avoid repeated individuals, a mutation also occurs when there is a crossover between equal individuals.

g) ***Population update****:* the new individuals are added to the current population, their fitness scores are evaluated and added to the fitness scores list of the current population. After that, the algorithm selects the best k individuals for the next generation, and excludes the worst k ones. The process repeats until the desired number of generations is reached.

Finally, at the end of the process, SiMOOD outputs the resulting population containing the final selected individuals. That is, the combinations that are candidates of being a source of hazards during simulation. However, these possible hazards cannot be confirmed until we perform the simulation tests. Next, we explain how we perform these simulations.

B. Simulation

Following the same term given in the previous subsection, in this part, we take the relevant individuals selected from the previous step and apply them to each frame of the simulation.

During simulation we collect the ground-truth from the original image provided by CARLA without perturbation, and we dynamically generate object's bounding boxes from the simulator data, adapting the method developed in [31]. This method generates 2D bounding boxes from the projected 3D bounding box of the *visible* objects in the camera image. It uses a distance filter, an angle filter, an occlusion filter, and instance segmentation of the LiDAR sensor provided by CARLA. Besides, to be as lightweight as possible, we adapted this module to generate the ground truth of the objects of interest in the safety scenario. For instance, in an advanced emergency braking scenario, one needs just the ground truth of the objects that interact with the ego-vehicle, such as cars and pedestrians, instead of checking for all other objects in the scene. All the ML model and safety results are stored for future analysis at the end of the simulation.

C. Evaluation

Besides simulation metrics such as processing time, and memory, we evaluate the safety-critical function (hazards) and the ML model (ML metrics). We use the number of hazards during simulation for the safety metrics. Here, we consider two types of hazards: a) accident (e.g., car hits pedestrian); and b) dangerous stop (e.g., the emergency brake is activated without necessity). For the ML metrics, we use the *mean average precision (mAP)* of the bounding boxes generated by the ML model [32]. It is a standard metric applied to evaluate object detectors. Next, we detail the experiments with SiMOOD.

IV. Experiments and results

We evaluate SiMOOD capacity to find a suitable choice of minimal perturbations that lead to hazards in CARLA simulations. We choose a safety-critical scenario in which a car uses an emergency braking system equipped with an ML-based object detection model (e.g., YOLO). In this scenario, a pedestrian crosses the street coming behind other objects. The ML model is responsible for detecting all objects that enter a safety-critical region in front of the ego-vehicle. Regarding the safety-critical region, the emergency braking system reacts to pedestrians that enter the region. The scenario is configured with CARLA scenario runner [33].

A. Experiment settings

Regarding the object detector algorithm, we use one of the most applied: YOLO v6 [13], a model based on stacked convolutional neural networks (CNN). The ML model was trained on the COCO dataset [34] with 10% of augmented data. We use the same augmentation techniques for generating noise and distributional shift perturbations in our approach. Worth to mention that we do not apply OOD perturbations during model training, just during test.

We applied 15 categories of OOD perturbations presented in [6], [35], [7], each one with its own levels of intensity ("no effect" included), totalling 175 different OOD perturbations.

TABLE I: OOD perturbation parameters.

OOD perturbation	**Levels of intensity**
Shifted pixels	11
Gaussian noise	8
Gaussian blur	24
Grid dropout	9
Coarse dropout	31
Channel dropout	3
Snow	6
Broken lens	11
Dirty	11
Condensation	11
Sun flare	11
Brightness	11
Contrast	11
Rain	6
Smoke	11

For novelty class experiments, we inject a fallen tree on a random road point since it is a familiar object in the operational design domain, but the ML model was not trained with this object in the training set. For anomaly experiments, we inject a vehicle overturned on a random point of the road since it is a known object to the ML model but in a way that is not expected to be found in the original dataset. More details can be found in our repository [12].

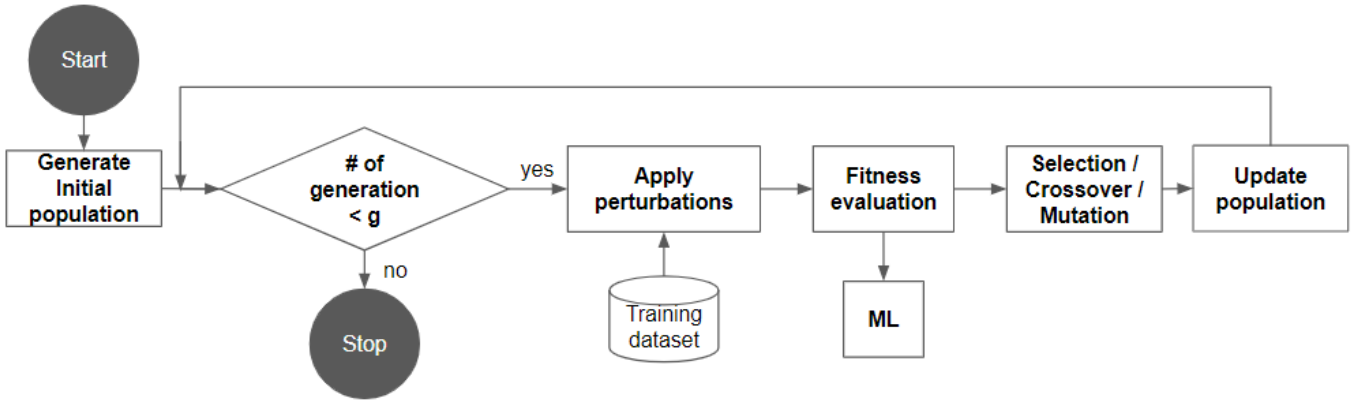


Fig. 3: Genetic algorithm approach: searching for relevant OOD perturbations.

Regarding the GA parameters, each individual has size $m = 2$. It is worth mentioning that one can change the size of the individual to $m = 1$ in case to find just the single perturbations that lead to hazards instead of a combination of them. Besides, the number of k selected individuals per generation is also fixed in $k = 2$, that is, we select the best pair of individuals per generation, replacing the worst ones. Worth to mention that k could be higher than 2 and we fix this value since this setting is a simple, and still traditional, approach [36] to validate our proposal. The mAP metric is applied as the ML metric. Therefore, the fitness score includes the worst mAP obtained from a given pair of perturbation.

Regarding the crossover and mutation probabilities, we follow the recommendations of Yang et al. [36] that showed from empirical results and theoretical studies that it is a good choice to set a relatively higher probability for crossover (e.g., in the range of 0.6 to 0.99). In contrast, the mutation probability can be very low (e.g., around 0.001 to 0.1). Therefore, we set our crossover probability as 0.99 and the mutation as 0.1. Finally, regarding the other GA parameters: number of generations, population size, and smoothness term ω were chosen after performing sensitivity analysis experiments.

B. Robustness of SiMOOD regarding its parameters

This subsection shows the robustness of SiMOOD regarding its parameters rather than performing a hyper-parameter optimization analysis. Moreover, for this analysis, we prioritize finding hazards with a reasonable amount of different perturbations rather than finding a huge number of hazards with repeated perturbations. That is, finding more hazards while having more diversity indicates better quality results.

We vary the number of generations g between $[10, 20, 30, 50]$, the initial population size n between $[10, 20, 30, 50]$, (each individual has 2 genes), and we also vary the value of ω between $[0.01, 0.1, 0.25, 0.5, 0.66, 0.75, 0.99]$. Next, we perform a diversity analysis, and a hazard analysis when varying the parameters of SiMOOD.

1) Diversity analysis on the influence of ω: The objective is to analyze how many unique individuals are generated when we change the smoothness term ω across the different amount of generations and population size. More unique individuals indicates more diversity in the results. Hence, as illustrated in Figure 4, the best value for better diversity tends to be $\omega = 0.5$. However, SiMOOD tends to be robust regarding this parameter, since we observe small but steady variations in the diversity in which $\omega \leq 0.5$ leads to slightly better results.

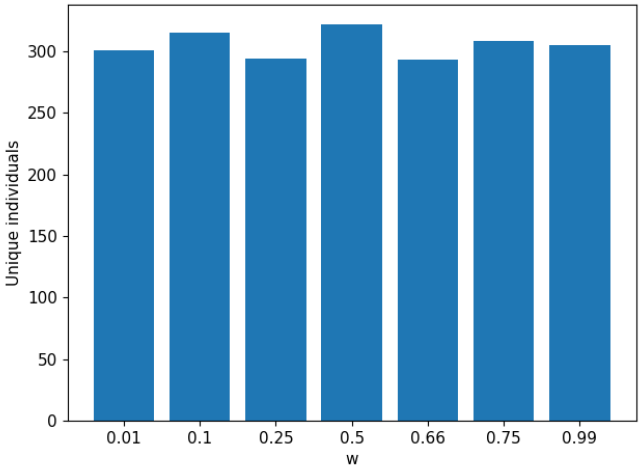


Fig. 4: Unique individuals per ω, generated across all variations of generations and population size (440 individuals).

It is worth mentioning that in the next analysis, we keep ω with the best value. The reason is that the amount of time to perform all simulation tests with different values of ω increases quite fast. That is, the simple range of values chosen for these analysis leads to 3080 different individuals, which would require 6,160 minutes of graphical simulation.

2) Hazard analysis on the influence of the population size and the number of generations when $\omega = 0.5$: Table II shows the number of unique genes, while Table III shows the number of hazards posteriorly found in the simulation. The best parameter values are the ones that lead to higher values

of unique genes (e.g., more diversity) and hazards. Best values of each column are marked with (*).

TABLE II: Number of unique genes in the selected population.

Generations	Population size			
	10	20	30	50
10	7 (35%)	16 (40%)	34 (57%)*	59 (59%)*
20	8 (40%)*	20 (50%)*	28 (47%)	48 (48%)
30	6 (30%)	5 (12%)	26 (43%)	42 (42%)
50	6 (30%)	7 (17%)	19 (31%)	32 (32%)

TABLE III: Number of hazards.

Generations	Population size			
	10	20	30	50
10	9 (90%)	11 (55%)	12 (40%)	12 (24%)
20	6 (60%)	20 (100%)*	23 (77%)	21 (42%)
30	10 (100%)*	20 (100%)*	30 (100%)*	25 (50%)
50	10 (100%)*	20 (100%)*	15 (50%)	23 (46%)

Tables II and III show that the number of unique genes increases gradually along with the population size when performed over low number of generations, but the number of hazards seems to be steady. Considering the best results in Table II and III, it seems that SiMOOD achieve a good compromise between diversity and hazards when performing with 20 generations g and population size $n = 20$. Besides, the number of predominant genes (e.g., repeated genes) increases with the number of generations. It indicates that SiMOOD does not need too many generations to find a good balance between diversity and uncovered hazards.

Regarding the variability of the experiments, we also run SiMOOD 10 times using the configuration that achieved the best values in both Tables (generation=20, population=20). Such experiments yielded standard deviation $\sigma = 2.5$ regarding the number of unique genes, and $\sigma = 5.5$ regarding the number of hazards, which is acceptable for our experiments.

C. Results and analysis

In this subsection, we show the results of two main experiments: single OOD perturbations that lead to hazards; and combinations of OOD perturbations that lead to hazards. These experiments were performed with the best parameters presented in the previous subsection: 20 generations g, population size $n = 20$, and $\omega = 0.5$.

1) Hazards provoked by single OOD perturbations: Figure 5 shows three examples.

The first sub-figure resulted from a vehicle crash due to a fallen tree not detected by the ML model. Since the safety-critical system depends on the ML model to "see" objects in the safety-critical region in front of the ego-vehicle, it does not trigger the brake-action, and the vehicle collides with the tree. In the second sub-figure, since the emergency braking system has the rule to react when a bounding box of a pedestrian intersects a warning region, the emergency braking system stops the vehicle in the middle of the street due to a "ghost" pedestrian detected by the ML model. Such a false detection was provoked by condensed water on the camera lens. Finally, in the Figure 5c), a new accident occurs; this time, the vehicle does not avoid a crash with a pedestrian not detected by the ML model when exposed to heavy smoke in the environment. Even though the ML model was capable of detecting the pedestrian, the detection was not made at the right time to avoid a crash after the brake-action was triggered. This case also shows the importance of testing on simulations instead of testing on static datasets.

2) Hazards provoked by combined OOD perturbations: Figure 6 shows an example of how the combination of different OOD perturbations can uncover new hazards. In the Figure 6c), we show a combination (smoke, 3) and (grid dropout, 1) found by SiMOOD that led to a collision between the vehicle and the pedestrian due to the failure of YOLO in detecting the pedestrian, leading to incorrect behavior of the emergency braking system. However, when one of these combinations happens alone (sub-figures a) and b)), the ML model can correctly detect the pedestrian, making the emergency braking system correctly trigger the brakes.

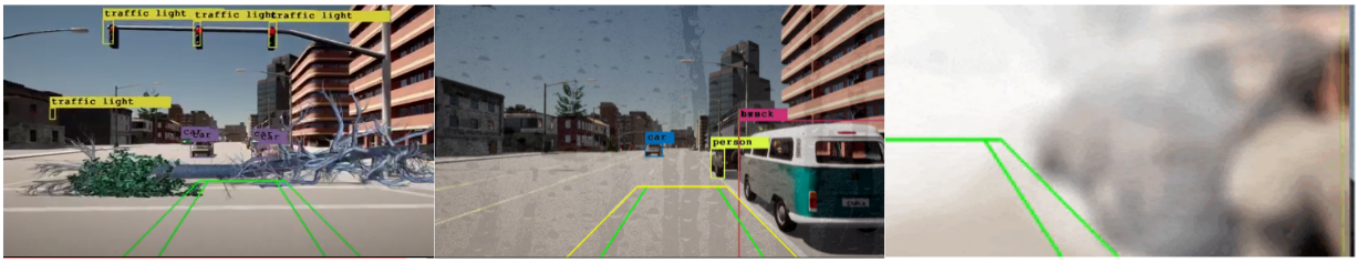
Simple combinations such as a sensor failure in a moment with a small amount of smoke or haze during runtime can be enough to provoke a hazard. Moreover, as illustrated in Figure 7, the order of the perturbations also matters. The reason is that same perturbations combined in a different order produce subtle differences in the image outcome, which is enough to hinder the performance of ML models. Next, we analyze the processing time and memory to run SiMOOD.

3) Processing time and memory usage: the experiments were performed in an Intel(R) i5-10500 CPU @ 3.10GHz, with 32GB of memory, six cores, and a GPU (Quadro RTX 4000). Below, we show time and memory analysis for the two parts of the approach: *a) generation, and b) simulation*.

a) Generation: SiMOOD can reduce the time necessary for finding hazards during the simulation due to its approach of performing the GA algorithm on data instead of applying it directly to the simulation. In our experiments, there are 175 possible perturbations T with their respective intensity levels, and the number of simultaneous perturbations for each combination (individual) is $m = 2$. Since the order of transformations matters, the set of possible combinations is given by $\frac{175!}{2!\times(175-2)!\times 2} = 30,450$ possible combinations.

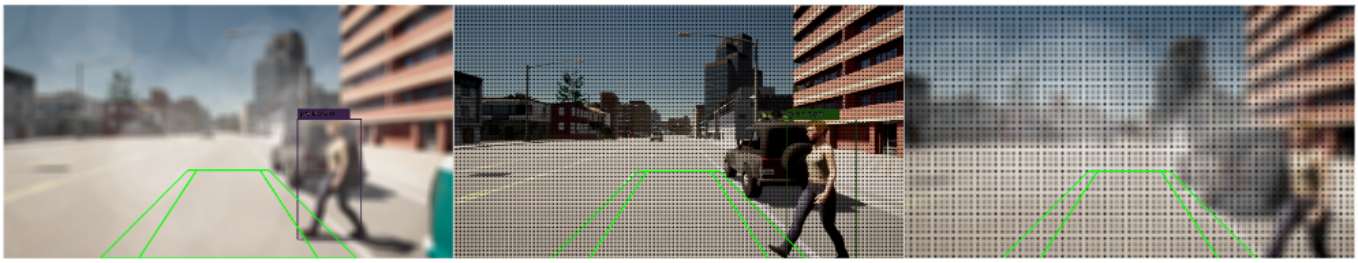
Besides, applying the GA directly into the simulation is suitable for the classes exposed during the test (in our case, pedestrians and cars). On the other hand, SiMOOD uses a GA-on-data approach, which considers all 80 classes in the COCO dataset. It allows SiMOOD to be extended to other types of scenarios and objects on it. Below, we provide a theoretical time analysis considering our GA and its parameters.

First, the GA starts with a time complexity of $O(nt)$ which n is the size of the initial population, and t is the amount of time to compute the fitness function. Second, for each subsequent generation, the GA will transform the k best



a) Novelty class b) Condensed water c) Heavy smoke

Fig. 5: Hazards uncovered by applying single OOD perturbations.
a) An accident due to a *unknown* object (tree) not detected by the ML model; **b)** A dangerous stop due to a false detection (ghost pedestrian) provoked by condensed water on the camera lens; **c)** An accident due to a *known* object (pedestrian) not detected by the ML model when exposed to heavy smoke on the environment.



a) Smoke (0.25) b) Grid dropout (2) c) Smoke(0.18) + Grid dropout(2)

Fig. 6: Hazards uncovered by combining two types of OOD perturbations.
a) YOLO detects a pedestrian in an environment with light smoke (intensity 0.25); **b)** YOLO detects a pedestrian despite a light grid dropout failure on the camera sensor (intensity 2); **c)** A collision happens due to a failure of YOLO on detecting a pedestrian when both conditions happen at the same time even with smoke transformation having a lower intensity than before.

(a) Grid dropout (1) + smoke (0.3). (b) Smoke (0.3) + grid dropout (1).

Fig. 7: Same OOD perturbations but in different order produce different image outcomes, which may uncover new hazards.
a) YOLO detects a pedestrian in the scenario with the OOD combination *grid dropout* + *smoke*. **b)** YOLO do not detect a pedestrian in the same scenario but with OOD perturbations in an inverted order (e.g., *smoke* + *grid dropout*).

individuals of the population with crossover and mutation operators, and posteriorly compute the fitness function, which adds $k * t * g$ to the processing time. Finally, the theoretical processing time when performing the GA is given by $nt+ktg$, and its required memory space is given by $m * k$.

Considering the aforementioned processing time formula, we calculate the expected time, *in seconds*, to find a population of hazards when *performing GA directly on the simulation*. Therefore, if we chose a simple configuration with an initial population $n = 20$ (0.0001% of the entire population), with 20 generations g, and the amount of time to render our scenario ($t = 120$), and the number of selected individuals per generation $k = 2$, then the expected processing time is given by $20 * 120 + 2 * 120 * 20 = 7,200$ seconds (*120 minutes*).

Regarding *performing the GA-on-data-first approach*, the time t to compute a fitness function is replaced by the time

spent generating the datasets (5 seconds per dataset) + the time applying the fitness function over each one of the generated datasets (7 seconds per dataset). That is, $t = 12$. Hence, the amount of time to perform GA over data is $20*12+2*12*20 = 720$ seconds. Finally, we add the time spent when running the simulation with the final population, that is, $120*20$. Therefore the total amount of time is given by $720 + 120 * 20 = 960$ seconds (*16 minutes*).

Besides, even SiMOOD reduced the processing time almost 10 times the GA-approach processing time, this difference could be even higher if we use a bit more complex scenarios such as longer scenarios with more objects to render, or if more than one different scenario needs to be tested.

b) Simulation: Table IV shows the processing time (seconds) and the memory usage (MB) *when performing the simulation* with and without using SiMOOD.

TABLE IV: Comparison of processing time and memory.

Time	**Time (with SiMOOD)**	**Overhead**
101.94	123.10	20.75%
Memory	**Memory (with SiMOOD)**	**Overhead**
3975.58	6708.09	68.73%

It is worth mentioning that the number of perturbations does not increase the average simulation time nor the total average memory used in the simulation. All perturbation functions are performed over a matrix with a fixed image size. However, since SiMOOD applies perturbations on high-resolution images (1280x720), it is necessary an extra amount of memory (2.7 GB) to perform the task without severely impacting the simulation speed. Therefore, SiMOOD requires a computer with 8GB of memory, which is quite normal for the current computational requirements. Moreover, SiMOOD can be optimized to perform better processing and consume less memory by performing parallelization and data compression.

D. Threats to validity

In order to analyze and mitigate threats to the validity of the results, we present below a summary of arguments for external and internal validation.

- *External validity*: Regarding the simulated sensor failures and weather conditions, there is always a gap between simulation and reality [37]. However, the tested OOD perturbations were already applied in several other papers from the literature with different levels of intensity.
  Regarding the chosen simulator to perform our experiments, it is worth mentioning that different simulators can yield different outcomes. However, CARLA simulator is an open-source simulator widely applied in the industry and in the literature. Besides, our approach can be extended for other simulators.

- *Internal validity*: What could have to lead us to the wrong conclusions in our study?

  a) Variability of simulated safety-critical scenarios: we tested our approach over a single but relevant safety-critical scenario. However, a safety-critical scenario can greatly vary depending on the expert needs. Therefore, it is not feasible to guarantee in a theoretical way that our approach will provide similar results across any safety-critical scenario. However, it is worth mentioning that this study is a first approach to this challenging problem. Thus, other scenarios can be added and tested in the future since our open-source code can easily be adapted/extended to new scenarios.
  b) Choice of OOD perturbation levels: despite the choice of perturbations on images capable of being interpreted by humans eyes, the amount of intensity considered as valid is a choice of the experts. Thus, the range of intensity over the perturbations can change the outcomes in different simulated scenarios.
  c) Choice of parameters of the GA: different values for the parameters of GA algorithm (e.g., number of selected candidates per generation, number of new generated individuals, crossover/mutation probability) can lead to different outcomes in both diversity and number of hazards in the simulation. However, we followed a traditional way to develop the GA approach and we also followed the recommendations from the literature [11].
  d) Choice of datasets for the ML model and for the GA: even though we chose a dataset widely applied in the computer vision literature, the choice of the amount of data accessible for training/validating and testing the ML model, and to perform the GA, can influence the fitness score and maybe also change the selected perturbations. However, we followed traditional ways to divide data (e.g., 80/20 for training and test).

V. Conclusion

In this work, we proposed SiMOOD, an evolutionary simulation testing for ML-based perception systems. SiMOOD is open-source [12] and is already integrated with CARLA, an open-source simulator for autonomous vehicles. SiMOOD was capable of finding out-of-distribution image perturbations that lead to hazards during simulation. Moreover, it was possible without performing such optimization algorithms loops directly into the simulation. Such strategy is able to save, up to 10 times, the amount of time needed to find a set of hazards.

We also showed that combinations of perturbations could expose different hazards. Moreover, these perturbations are not commutative. Hence, the order of occurrence of such perturbations can also uncover new hazards in the simulation.

A limitation of this work is that we tested just one type of state-of-the-art object detector (e.g.,YOLO). Other ML models should be added to give more robust results regarding the applicability of the proposed approach. Another limitation of this work is the additional memory usage added to the simulation. However, this is the first version of SiMOOD, and it can be improved in the next versions since it is open to the community. As a next step, we intend to add a

scenario generation process similar to Abdessalem et al. [38], capable of varying the scenarios, their objects and parameters. Besides, new safety-oriented metrics [39] can also be added to SiMOOD, which can produce more types of analysis.

REFERENCES

[1] D. Bogdoll, M. Nitsche, and J. M. Zöllner, "Anomaly detection in autonomous driving: A survey," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022, pp. 4488–4499.

[2] Y. Sun, C. Guo, and Y. Li, "React: Out-of-distribution detection with rectified activations," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[3] M. Sabokrou, M. Khalooei, M. Fathy, and E. Adeli, "Adversarially learned one-class classifier for novelty detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3379–3388.

[4] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *International conference on learning representations (ICLR), New Orleans, United States*, 2019.

[5] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *International conference on machine learning (ICML), New York, United States*, 2016, pp. 1050–1059.

[6] F. Secci and A. Ceccarelli, "On failures of rgb cameras and their effects in autonomous driving applications," in *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2020, pp. 13–24.

[7] R. S. Ferreira, J. Arlat, J. Guiochet, and H. Waeselynck, "Benchmarking safety monitors for image classifiers with machine learning," *26th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2021), Perth, Australia*, 2021.

[8] 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*, 2017, pp. 1–16.

[9] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," in *International conference on learning representations*, 2017.

[10] A. Shafaei, M. Schmidt, and J. J. Little, "A less biased evaluation of out-of-distribution sample detectors," *30th British Machine Vision Conference, Cardiff, Wales*, 2019.

[11] S. Mirjalili, "Genetic algorithm," in *Evolutionary algorithms and neural networks*. Springer, 2019, pp. 43–55.

[12] R. S. Ferreira, "Code repository for the paper: SiMOOD: evolutionary safety simulation testing with out-of-distribution images," 2022. [Online]. Available: https://github.com/raulsenaferreira/SiMOOD

[13] G. Jocher, A. Stoken, A. Chaurasia, J. Borovec, NanoCode, TaoXie, Y. Kwon, K. Michael, L. Changyu, J. Fang, A. V, Laughing, tkianai, yx NONG, P. Skalski, A. Hogan, J. Nadar, imyhxy, L. Mammana, A. Wang, C. Fati, D. Montes, J. Hajek, L. Diaconu, M. T. Minh, Marc, albinxavi, fatih, oleg, and wang haoyang, "Yolov5: v6.0," [Online; accessed 21-January-2022]. [Online]. Available: https://doi.org/10.5281/zenodo.5563715

[14] P. Perera and V. M. Patel, "Deep transfer learning for multiple class novelty detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), California, United States*, 2019, pp. 11 544–11 552.

[15] R. Stumpf, "Autopilot blamed for tesla's crash into overturned truck," 2020, [Online; accessed 21-June-2022]. [Online]. Available: https://www.thedrive.com/news/33789/autopilot-blamed-for-teslas-crash-into-overturned-truck

[16] Z. Shen, J. Liu, Y. He, X. Zhang, R. Xu, H. Yu, and P. Cui, "Towards out-of-distribution generalization: A survey," *arXiv preprint arXiv:2108.13624*, 2021.

[17] R. S. Ferreira, G. Zimbrão, and L. G. Alvim, "Amanda: Semi-supervised density-based adaptive model for non-stationary data with extreme verification latency," *Information Sciences*, vol. 488, pp. 219–237, 2019.

[18] 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)*. IEEE, 2020, pp. 25–36.

[19] E. Zapridou, E. Bartocci, and P. Katsaros, "Runtime verification of autonomous driving systems in carla," in *International Conference on Runtime Verification*. Springer, 2020, pp. 172–183.

[20] V. Singh, S. K. S. Hari, T. Tsai, and M. Pitale, "Simulation driven design and test for safety of ai based autonomous vehicles," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 122–128.

[21] P. Arcaini, A. Calò, F. Ishikawa, T. Laurent, X.-Y. Zhang, S. Ali, F. Hauer, and A. Ventresque, "Parameter-based testing and debugging of autonomous driving systems," in *2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*. IEEE, 2021, pp. 197–202.

[22] M. O'Kelly, A. Sinha, H. Namkoong, J. Duchi, and R. Tedrake, "Scalable end-to-end autonomous vehicle testing via rare-event simulation," in *Advances in neural information processing systems (NeurIPS)*, 2018.

[23] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," in *proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 1–18.

[24] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th international conference on software engineering*, 2018, pp. 303–314.

[25] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems," in *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2018, pp. 132–142.

[26] 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, no. 5, pp. 1–30, 2021.

[27] T. Dreossi, A. Donzé, and S. A. Seshia, "Compositional falsification of cyber-physical systems with machine learning components," *Journal of Automated Reasoning*, vol. 63, no. 4, pp. 1031–1053, 2019.

[28] G. Rossolini, F. Nesti, G. D'Amico, S. Nair, A. Biondi, and G. Buttazzo, "On the real-world adversarial robustness of real-time semantic segmentation models for autonomous driving," *arXiv preprint arXiv:2201.01850*, 2022.

[29] A. Boloor, K. Garimella, X. He, C. Gill, Y. Vorobeychik, and X. Zhang, "Attacking vision-based perception in end-to-end autonomous driving models," *Journal of Systems Architecture*, vol. 110, p. 101766, 2020.

[30] H. Fahmy, F. Pastore, and L. Briand, "Hudd: A tool to debug dnns for safety analysis," in *2022 IEEE/ACM 44st International Conference on Software Engineering*, 2022.

[31] M. Adib, "CARLA 2D Bounding Box Annotation Module," https://github.com/MukhlasAdib/CARLA-2DBBox/, 2022, [Online; accessed 23-June-2022].

[32] J. Cartucho, R. Ventura, and M. Veloso, "Robust object recognition through symbiotic deep learning in mobile robots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 2336–2341.

[33] NVidia, "CARLA scenario runner," https://carla-scenariorunner.readthedocs.io/en/latest/, 2022, [Online; accessed 23-June-2022].

[34] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[35] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, "Albumentations: fast and flexible image augmentations," *Information*, vol. 11, no. 2, p. 125, 2020.

[36] X.-S. Yang, "Chapter 5 - genetic algorithms," in *Nature-Inspired Optimization Algorithms*, X.-S. Yang, Ed. Oxford: Elsevier, 2014, pp. 77–87.

[37] A. Stocco, B. Pulfer, and P. Tonella, "Mind the gap! a study on the transferability of virtual vs physical-world testing of autonomous driving systems," *arXiv preprint arXiv:2112.11255*, 2021.

[38] R. B. 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)*. IEEE, 2018, pp. 1016–1026.

[39] J. Guerin, R. S. Ferreira, K. Delmas, and J. Guiochet, "Unifying evaluation of machine learning safety monitors," *arXiv preprint arXiv:2208.14660*, 2022.