SafeBench: A Benchmarking Platform for Safety Evaluation of Autonomous Vehicles

Chejian Xu1, Wenhao Ding2, Weijie Lyu1, Zuxin Liu2, Shuai Wang2, Yihan He2, Hanjiang Hu2, Ding Zhao2, Bo Li1
1University of Illinois at Urbana-Champaign 2Carnegie Mellon University
{chejian2,wlyu3,lbo}@illinois.edu, dingzhao@cmu.edu
{wenhaod,zuxinl,shuaiwa2,yihanhe,hanjianh}@andrew.cmu.edu

Abstract

As shown by recent studies, machine intelligence-enabled systems are vulnerable to test cases resulting from either adversarial manipulation or natural distribution shifts. This has raised great concerns about deploying machine learning algorithms for real-world applications, especially in the safety-critical domains such as autonomous driving (AD). On the other hand, traditional AD testing on naturalistic scenarios requires hundreds of millions of driving miles due to the high dimensionality and rareness of the safety-critical scenarios in the real world. As a result, several approaches for autonomous driving evaluation have been explored, which are usually, however, based on different simulation platforms, types of safety-critical scenarios, scenario generation algorithms, and driving route variations. Thus, despite a large amount of effort in autonomous driving testing, it is still challenging to compare and understand the effectiveness and efficiency of different testing scenario generation algorithms and testing mechanisms under similar conditions.

In this paper, we aim to provide the first unified platform SafeBench to integrate different types of safety-critical testing scenarios, scenario generation algorithms, and other variations such as driving routes and environments. In particular, we consider 8 safety-critical testing scenarios following National Highway Traffic Safety Administration (NHTSA) and develop 4 scenario generation algorithms considering 10 variations for each scenario. Meanwhile, we implement 4 deep reinforcement learning-based AD algorithms with 4 types of input (e.g., bird’s-eye view, camera) to perform fair comparisons on SafeBench. We find our generated testing scenarios are indeed more challenging and observe the trade-off between the performance of AD agents under benign and safety-critical testing scenarios. We believe our unified platform SafeBench for large-scale and effective autonomous driving testing will motivate the development of new testing scenario generation and safe AD algorithms. SafeBench is available at https://safebench.github.io.

1 Introduction

Innovations driven by recent progress in machine learning (ML) have shown human-competitive performance in sensing [1], decision-making [2], and manipulation [3]. However, several studies have shown that when such powerful ML models are exposed to adversarial attacks they can be fooled, evaded, and misled in ways that would have profound security implications: image recognition, natural language processing, and audio recognition systems have all been attacked [4, 5, 6, 7]. As ML-based models and approaches have expanded to real-world safety-critical applications, such as Autonomous Driving (AD), the question of safety is becoming a crux for the transition from theories...
Challenges. Despite the great importance of safety evaluation for AD algorithms, it is challenging
to comprehensively and quantitatively evaluate AD algorithms due to both real-world data and
evaluation design challenges. First, in practice, the safety-critical driving scenarios are “rare” – can
be found by driving every 30,000 miles [11], which leads to the fact that current AD testing requires
driving millions of miles with large economic and environmental costs. In addition, such rarity
also requires the evaluation methods to have an accelerated feature with a probabilistic convergence
guarantee to avoid being over-optimistic. Previous work [12] solve this problem for abstract
simple models by using large deviation theories such as importance sampling (IS) and cross entropy
(CE) [4]. However, these approaches are shown to have reached bottlenecks when dealing with
ML algorithms with increasing complexity. In fact, recent studies [13] have shown that these
classical IS/CE based approaches and tools may consistently underestimate the risk when dealing
with complex systems. Moreover, such peril has been identified in different evaluation approaches
[16] [17] [18] [19] [20] [21], which have been already adopted by industry [22] and test agencies [23] in
the U.S. to assess the safety of AVs. Second, although several learning-based scenario generation
approaches are later proposed to overcome the above challenge [24] [25] [26] [27], existing evaluation
tools and platforms are usually based on their own design, such as dataset selection, safety-critical
scenario definition and generation, evaluation metrics, and input types. This makes it very challenging
to fairly compare different AD algorithms or interpret different evaluation results.

In this paper, we focus on designing and developing the first unified robustness and safety evaluation
platform for AD algorithms, SafeBench. In particular, we design SafeBench based on the open-
sourced simulation platform Carla [28]. SafeBench consists of 4 modules, including Agent Node, Ego
Vehicle, Scenario Node, and Evaluation Node. Based on our platform, we systematically evaluate
the AD algorithms on 2,352 generated safety-critical testing scenarios, such as Straight Obstacle
and Lane Changing together with other benign scenarios. For each safety-critical scenario, we
implement 4 scenario generation algorithms for comparison. In addition, for each scenario, we
select 10 diverse driving routes to ensure the generalization of our evaluation results. We report the
evaluation results based on 10 metrics, such as collision rate, frequency of running red lights, and
average percentage of route completion. Finally, we developed 4 reinforcement learning-based AD
algorithms with different perceptual capabilities on SafeBench. Specifically, we provide 4 input
types, ranging from low-dimensional state representations to complicated visual inputs. Based on
our comprehensive evaluation, we find that (1) there is a performance trade-off for different AD
algorithms under benign and safety-critical scenarios, (2) some safety-critical scenarios have higher
transferability across AD algorithms, (3) different scenario generation algorithms achieve different
levels of effectiveness even when generating the same scenario, (4) different AD algorithms achieve
advantages over others under different metrics. Our findings suggest that testing AD algorithms on
high-quality safety-critical scenarios is necessary and can largely improve testing efficiency, and we
should consider a combination of testing scenarios and generation algorithms for effective testing.

Contributions. In this work, we aim to provide the first unified evaluation platform for different AD
algorithms by generating diverse safety-critical scenarios with different generation algorithms and
evaluation metrics. Our evaluation platform SafeBench includes the following properties.

- Unified benchmarking platform with modularized design. Our evaluation platform consists of
  4 modules, including Ego vehicle, Agent, Scenario, and Evaluation. It is also flexible to replace,
  add, or delete modules for future functionalities and evaluations.
- Comprehensive coverage of safety-critical scenario generation. In SafeBench, we have inte-
grated 2,352 testing scenarios, which have provided comprehensive coverage of known safety-
critical scenarios in the real world, and it is flexible to add more testing scenarios by applying
generation methods on new template scenarios.
Table 1: Comparison of Evaluation Platform

<table>
<thead>
<tr>
<th>Simulator</th>
<th>Safety-critical Scenarios</th>
<th>Number of Scenario</th>
<th>Realistic Perception</th>
<th>Customized Scenario</th>
<th>Backend</th>
<th>Baselines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safebench</td>
<td>✓</td>
<td>3,000+</td>
<td>✓</td>
<td>✓</td>
<td>CARLA</td>
<td>✓</td>
</tr>
<tr>
<td>Scenario Runner</td>
<td>✓</td>
<td>10</td>
<td>✓</td>
<td>✓</td>
<td>CARLA</td>
<td></td>
</tr>
<tr>
<td>DI-Drive Casezoo</td>
<td>✓</td>
<td>8</td>
<td>✓</td>
<td>✓</td>
<td>CARLA</td>
<td></td>
</tr>
<tr>
<td>SUMMIT</td>
<td>×</td>
<td>N/A</td>
<td>✓</td>
<td>×</td>
<td>UE4</td>
<td>✓</td>
</tr>
<tr>
<td>Scenario Studio</td>
<td>×</td>
<td>N/A</td>
<td>×</td>
<td>✓</td>
<td>SMARTS</td>
<td>✓</td>
</tr>
<tr>
<td>CommonRoad</td>
<td>×</td>
<td>N/A</td>
<td>✓</td>
<td>✓</td>
<td>None</td>
<td>×</td>
</tr>
<tr>
<td>CausalCity</td>
<td>×</td>
<td>N/A</td>
<td>✓</td>
<td>✓</td>
<td>UE4</td>
<td>×</td>
</tr>
<tr>
<td>MetaDrive</td>
<td>×</td>
<td>6</td>
<td>✓</td>
<td>✓</td>
<td>Panda3D</td>
<td>✓</td>
</tr>
<tr>
<td>highway-env</td>
<td>×</td>
<td>6</td>
<td>✓</td>
<td>✓</td>
<td>None</td>
<td>×</td>
</tr>
<tr>
<td>SUMO NETEDIT</td>
<td>×</td>
<td>N/A</td>
<td>✓</td>
<td>✓</td>
<td>SUMO</td>
<td>×</td>
</tr>
<tr>
<td>SimMobilityST</td>
<td>×</td>
<td>N/A</td>
<td>✓</td>
<td>✓</td>
<td>None</td>
<td>×</td>
</tr>
<tr>
<td>L2R</td>
<td>×</td>
<td>3</td>
<td>✓</td>
<td>✓</td>
<td>UE4</td>
<td>×</td>
</tr>
<tr>
<td>AutoDRIVE</td>
<td>×</td>
<td>N/A</td>
<td>✓</td>
<td>✓</td>
<td>Unity</td>
<td>×</td>
</tr>
<tr>
<td>Deepdrive</td>
<td>×</td>
<td>N/A</td>
<td>✓</td>
<td>✓</td>
<td>UE4</td>
<td>×</td>
</tr>
<tr>
<td>esmini</td>
<td>×</td>
<td>N/A</td>
<td>✓</td>
<td>✓</td>
<td>Unity</td>
<td>×</td>
</tr>
<tr>
<td>AutonoViSim</td>
<td>×</td>
<td>N/A</td>
<td>✓</td>
<td>✓</td>
<td>PhysX</td>
<td>×</td>
</tr>
</tbody>
</table>

- **Comprehensive coverage of scenario generation algorithms.** For each testing safety-critical scenario, we developed 4 generation algorithms, so that we are able to evaluate AD safety on the scenario level, but also on the generation algorithm level.

- **Diverse metrics on safety measurement of different AD algorithms.** We report our evaluation based on 10 evaluation metrics, based on three levels: safety, functionality, and etiquette.

- **General leaderboard of safety evaluation and extensible findings.** We provide a comprehensive leaderboard for the robustness and safety evaluation of 4 AD algorithms, and we observe different performances of these AD algorithms under different controllable settings.

- **High flexibility and effectiveness.** Our evaluation platform is flexible to be integrated into other simulation platforms and different devices. Once the AD algorithm is trained, it is very effective to be tested on our generated testing scenarios.

2 Related work

Existing AD algorithm evaluation approaches and platforms can be categorized into three types based on how the testing driving scenarios are generated. First, the **data-driven** based generation and testing approaches [44, 45, 46, 47] focus on real-world data sampling and distribution density estimation. This line of research is able to model the real-world driving conditions, while requiring a large number of data collection to capture the “rare” safety-critical scenarios for testing. Second, the **adversary-based** generation and testing approaches [48, 49, 50] model the surrounding agents (e.g., vehicles and pedestrians) as adversarial agents to generate safety-critical driving scenarios. Third, the **knowledge-based** generation and testing approaches [51, 52] aim to integrate domain knowledge such as traffic rules as additional constraints to guide the testing scenario generation process. Recently, the latter two categories have shown efficient and effective evaluation results under specific driving environments and settings, and therefore we mainly focus on them in this work. However, existing driving scenario generation and testing approaches are developed on different platforms with different AD algorithms and sensor configurations, etc., making it challenging to directly compare the effectiveness of different testing scenarios, scenario generation algorithms, and the safety of AD algorithms. Thus, in this work we will provide the first unified platform SafeBench, to generate safety-critical scenarios with different algorithms considering a range of environments and configurations for fair comparison based on a comprehensive set of evaluation metrics. In addition, several works have been conducted to test the safety of autonomous vehicles from the software testing perspective [53], which mainly focuses on identifying the safety violations from the software level. Such testing frameworks can be integrated into SafeBench as well for comprehensive testing.

**Comparison with other AD evaluation platforms** To accurately posit our SafeBench platform in the AD evaluation area, we summarize existing platforms developed for autonomous vehicle
In this section, we will first provide an overview of our platform SafeBench, followed by the details of our developed scenario generation algorithms and variants, as well as the evaluation metrics.

### 3.1 Platform structure

#### Overview

In Figure 1, we show the structure of our SafeBench platform. This platform runs in the Docker container and is built upon the Carla simulator. We use ROS for communication between the modules in the platform. In particular, SafeBench consists of 4 components (nodes) as introduced in the following.

- **Ego vehicle** provides a virtual vehicle including the configurations of sensors (e.g., the positions and parameters of LiDAR, Camera, and Radar), the global planner, and the appearance of the vehicle. The testing AD algorithms are deployed in this node to interact with the driving scenarios. Users can change the configuration of this node to satisfy the requirement of their algorithms.

- **Agent node** is designed to train and manage AD algorithms for ego and surrounding vehicles, taking as input the observation information from the testing scenarios and outputting the controlling signals. AD algorithms managed by this node can be trained on our platform.

- **Scenario node** is the core part of SafeBench, which is responsible for organizing and generating testing scenarios. These scenarios control the behaviors of traffic participants (e.g., pedestrians and surrounding vehicles) and static driving environments (e.g., road layout and status of traffic lights).

- **Evaluation node** is designed to provide comprehensive evaluations by testing different AD algorithms under diverse generated driving scenarios based on different metrics. The Evaluation Node collects all information during testing and provides an evaluation summary on different levels.

### 3.2 Safety-critical testing scenarios

In this section, we first define the safety-critical traffic testing scenarios we considered in this work, containing 8 most representative and challenging driving scenarios of pre-crash traffic summarized by the National Highway Traffic Safety Administration (NHTSA). In addition, for each scenario, we design ten diverse driving routes that vary in terms of surrounding environments, number of lanes, road signs, etc. Please see more detailed scenario definitions and route variants in Appendix A.3.

#### Pre-crash safety-critical scenarios

We show the 8 pre-crash scenarios in the right part of Figure 1. In each scenario, the ego vehicle needs to drive along a pre-defined route and react to emergencies that occur on the road while driving. Throughout the process, the ego vehicle should follow the traffic rules and avoid potential car accidents.
Driving routes. In practice, a driving scenario may involve many variants. For instance, small changes in the vehicle location or in the surrounding environment may lead to big changes in vehicle decision-making. In order to provide a more comprehensive safety evaluation, we design 10 driving routes for each safety-critical scenario. Each driving route has a sequence of pre-defined waypoints. Different driving routes of the same scenario may have a different number of lanes, different scenes (e.g., intersections, T-junctions, bridges, etc.), or different road signs, which restrict the vehicle behaviors in different ways. We show 2 example route variants of Turning Obstacle in Figure 2.

3.3 Safety-critical scenario generation algorithms

In this section, we detail how we collect and optimize safety-critical testing scenarios using different generation algorithms. Specifically, for each driving route mentioned above, we develop 4 algorithms to generate various testing samples. These generation algorithms mainly fall into two categories: adversary-based generation and knowledge-based generation.

3.3.1 Adversary-based generation

The state-of-the-art adversarial generation algorithms usually consist of two components: the scenario generator, and the victim model (i.e., the ego vehicle or tested AD agent). Existing adversarial generation frameworks adopt different strategies to manipulate traffic scenarios, such as perturbing the position of surrounding vehicles (SVs) or forcing a cyclist to take an adversarial action, such that the victim model will crash into SVs and fail in the generated scenario. To examine the safety and robustness of the tested AD agent against such adversarial scenarios, we select two representative algorithms as follows: (i) Learning-to-collide (LC) is a black-box algorithm that optimizes the initial poses of a cyclist to attack the AD algorithm. Following the default setting, we formulate the traffic scenarios as a series of auto-regressive building blocks and obtain the generated scenarios by sampling from the joint distribution of these blocks. The policy gradient method REINFORCE is used to solve the scenario optimization problem. In LC, the authors only focus on generating Turning Obstacle scenario, so we adapt the method to all the 8 scenarios and generate different initial conditions for all the driving routes. (ii) AdvSim (AS) directly manipulates existing trajectories to perturb the driving paths of SVs, posing dangers to the tested AD agent. We follow the default setting and use the kinematic bicycle model to represent and calculate the full trajectory of SVs. Based on the results obtained by interacting with the driving environment, we optimize the trajectory parameters using the black-box search algorithm Bayesian Optimization. Similarly, in our experiments, we generate adversarial trajectories for all the route variants.

3.3.2 Knowledge-based generation

In the physical world, driving scenarios need to satisfy traffic rules and physical laws. Scenarios generated by adversarial algorithms, however, sometimes violate these rules. Therefore, we develop novel generation algorithms that integrated domain knowledge into the generation process. We select two representative algorithms as follows. (i) Carla Scenario Generator (CS) is a module built on the Carla Simulator which uses rule-based methods to construct testing scenarios. Following the standard process, we adopt the rules and use grid search to generate safety-critical scenario parameters for all the 8 traffic scenarios. (ii) Adversarial Trajectory Optimization (AT) uses explicit knowledge as constraints to guide the scenario optimization process. We adopt the same constraints that needed to be satisfied and use the default PSO-based blackbox optimization for generating all kinds of testing scenarios in SafeBench.

3.4 Evaluation metrics

In this section, we introduce the evaluation metrics used in SafeBench. Specifically, we evaluate the performance of AD algorithms on 3 levels: Safety level, Functionality level, and Etiquette level.
Within each level, we design several metrics focusing on different aspects. Finally, an overall score is calculated as a weighted sum of all the evaluation metrics introduced below.

**Safety level** To evaluate the safety of given AD algorithms, we follow existing works and consider 4 evaluation metrics focusing on serious violations of traffic rules: collision rate (CR), frequency of running red lights (RR), frequency of running stop signs (SS), and average distance driven out of road (OR). Formally, we define the scenario trajectory as $\tau$, which is sampled from a scenario distribution $P$, then the collisions happened in one scenario after testing the AD algorithm can be represented as $c(\tau)$. Similarly, we obtain the number of running red lights $r(\tau)$, running stop signs $s(\tau)$, and distance driven out of road $d(\tau)$. The 4 metrics are concretely calculated as:

$$CR = \mathbb{E}_{\tau \sim P}[c(\tau)], \quad RR = \mathbb{E}_{\tau \sim P}[r(\tau)], \quad SS = \mathbb{E}_{\tau \sim P}[s(\tau)], \quad OR = \mathbb{E}_{\tau \sim P}[d(\tau)].$$

**Functionality level** In each testing scenario, the AD agent is expected to follow and complete a specific route. This level of evaluation metrics is used to measure the functional ability of AD agents to finish such a task. Inspired by previous works, we develop 3 metrics as follows: route following stability (RF), average percentage of route completion (Comp), and average time spent to complete the route (TS). To calculate RF, we use the average distance between the ego vehicle and the reference route during each testing $x(\tau)$. Then we calculate $RF = 1 - \mathbb{E}_{\tau \sim P}[\min\{\frac{x(\tau)}{x_{\max}}, 1\}]$, where $x_{\max}$ is a constant indicating the maximum deviation distance. Comp is calculated as $Comp = \mathbb{E}_{\tau \sim P}[p(\tau)]$, where $p(\tau)$ is the percentage of route completion of each testing scenario. TS is the average time spent for completing the routes successfully: $TS = \mathbb{E}_{\tau \sim P}[t(\tau)]|p(\tau) = 100\%|$, where $t(\tau)$ denotes the time cost of each testing scenario.

**Etiquette level** In practice, driver etiquette is an indicator of the driving skills of AD algorithms. Here we follow existing works and consider 3 metrics accordingly: average acceleration (ACC), average yaw velocity (YV), and frequency of lane invasion (LI). Similarly, these metrics are calculated as the expectation over all testing scenarios: $ACC = \mathbb{E}_{\tau \sim P}[acc(\tau)], \quad YV = \mathbb{E}_{\tau \sim P}[y(\tau)], \quad LI = \mathbb{E}_{\tau \sim P}[l(\tau)]$, where $acc(\tau), y(\tau), l(\tau)$ denote the accelerations, yaw velocities, and number of lane invasions respectively.

**Overall score** To obtain an evaluation overview of the quality of AD algorithms, we aggregate all the metrics and report an overall score (OS), which is a weighted sum of the 10 metrics introduced above. Specifically, the overall score is calculated as: $OS = \sum_{i=1}^{10} w_i \times g(m^i)$, where $m^i$ is the $i^{th}$ metric, $w_i$ is the corresponding weight, $g(m^i)$ is defined as

$$g(m^i) = \begin{cases} \frac{m^i}{m_{\max}^i}, & m^i \text{ is the higher the better} \\ 1 - \frac{m^i}{m_{\max}^i}, & m^i \text{ is the lower the better} \end{cases}$$

where $m_{\max}^i$ is a constant indicating the maximum allowed value of $m^i$. More details of the constant, parameters, and weight selection are in Appendix A.4.

### 4 Benchmark evaluation on SafeBench

In this section, we will first introduce the AD algorithms we will test which are based on different input state types, then illustrate our testing scenario generation and selection details, followed by our comprehensive benchmark results and corresponding observations and findings.

#### 4.1 AD algorithms tested on SafeBench

We test various types of algorithms based on the safety-critical scenarios in SafeBench. We particularly focus on reinforcement learning-based self-driving methods, since they require minimum domain knowledge of the overall system and driving scenarios. One only needs to specify the reward function, action space, and state space, then train the agent by interacting with the scenario, and finally obtain a self-driving agent with reasonable performance. The reward function is given by a linear combination of the route following bonus, the collision penalty, the speeding penalty, and the energy consumption penalty. The action space is specified by the steering and throttle of the vehicle.

We select 4 representative deep RL methods for evaluation, including a stochastic on-policy algorithm – Proximal Policy Optimization (PPO), a stochastic off-policy method – Soft Actor Critic.
(SAC) and two deterministic off-policy approaches – Deep Deterministic Policy Gradient (DDPG) and Twin Delayed DDPG (TD3). To encourage the diversity of evaluation agents, we vary the state space to equip them with different perceptual capabilities. We design 4 state spaces for each RL algorithm based on previous works as follows. The detailed model design and hyperparameters are presented in appendix A.5.

- **4D**: The basic observation type contains only 4 dimensions of observation: distance to the waypoint, longitude speed, angular speed, and a front-vehicle detection signal.

- **4D+Dir**: For a more complex observation type, we add another 7 dimensions of observations, which are "Command (turn left, turn right or go straight)" and vectors that represent the direction of the ego vehicle, current waypoint, and target waypoint.

- **4D+BEV**: We render the ego vehicle’s local semantic map using information provided by CARLA as the bird’s-eye view (BEV) image, where the vehicles are represented by boxes. Lanes and routes are represented by line segments. We incorporate the BEV image together with 4 dimensional states to form this observation type.

- **4D+Cam**: This observation type includes an image captured by the front camera with 4D.

### 4.2 Driving scenarios for testing

#### Scenario generation.
We apply 4 safety-critical scenario generation algorithms to 8 template scenarios, each of which contains 10 diverse driving routes. For each generation algorithm, we keep 9 or 10 testing scenarios based on their qualities. Thus, in total, we generate 3,140 testing scenarios for evaluation. We note that some scenario generation algorithms require a surrogate model to search for effective safety-critical configurations. For instance, we follow the setup of LC to train a surrogate SAC model based on random benign scenarios.

#### Scenario selection.
After collecting the raw testing scenarios, we select scenarios with desired properties. Specifically, we test all the generated scenarios on 4 AD algorithms with basic observation type and select scenarios that cause the most collisions. In Figure 3, we show a histogram of the distribution for collisions. We only keep scenarios that cause collisions for at least 2 algorithms during testing, which is shown in red in Figure 3. The selected testing scenarios have high transferability across AD algorithms and high risk levels, which further improves both the effectiveness and efficiency of AD evaluation. After the selection, we obtain 2,352 testing scenarios in total. More details can be found in Appendix A.1.

#### Analysis of generation algorithms and testing scenarios.
We analyze the properties of scenario generation algorithms based on a range of metrics, including the collision rate (CR), overall score (OS), and the overall selection rate (SR) for each scenario before and after selection. As shown in Table 2, first, the scenario selection process indeed helps to improve CR of the testing scenarios to induce more safety-critical ones: with the highest improvement as 30% for LC. Second, AT is the most effective algorithm to cause both high CR and low OS. In fact, 73.4% of the generated scenarios by AT can cause collisions to the surrogate model and it will increase to 81.1% after scenario selection. The scenarios generated by AT achieve OS as 0.546 and it will further decrease to 0.508 after scenario selection, indicating its testing effectiveness. Third, regarding the overall SR of different algorithms, scenarios generated by CS achieve the highest SR, which means CS is the best algorithm in terms of transferability across different AD algorithms. Specifically, 85.5% of scenarios generated by CS can successfully cause collisions to other unseen AD agents. Finally, among different scenarios, Vehicle Passing is the most difficult with the highest CR and lowest OS.
Table 2: Statistics of scenario generation/selection. We report collision rate (CR) before and after scenario selection (S-CR) to measure the effectiveness of different scenario generation algorithms. The overall score (OS) before and after scenario selection (S-OS) are used to demonstrate the safety-critical scenario generation capability of different algorithms. The selection rate (SR) is reported to evaluate the transferability of generation algorithms across AD agents. The last column shows the average over all the scenarios, with bold numbers indicating the best performance among the 4 generation algorithms. LC: Learning-to-collide, AS: AdvSim, CS: Carla Scenario Generator, AT: Adversarial Trajectory Optimization. ↑/↓: higher/lower the better.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CR ↑</td>
<td>LC</td>
<td>0.320</td>
<td>0.140</td>
<td>0.560</td>
<td>0.920</td>
<td>0.410</td>
<td>0.630</td>
<td>0.458</td>
<td>0.470</td>
<td>0.489</td>
</tr>
<tr>
<td></td>
<td>AS</td>
<td>0.570</td>
<td>0.350</td>
<td>0.650</td>
<td>0.900</td>
<td>0.600</td>
<td>0.820</td>
<td>0.520</td>
<td>0.550</td>
<td>0.620</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>0.610</td>
<td>0.630</td>
<td>0.322</td>
<td>0.900</td>
<td>0.767</td>
<td>0.756</td>
<td>0.667</td>
<td>0.711</td>
<td>0.670</td>
</tr>
<tr>
<td></td>
<td>AT</td>
<td>0.680</td>
<td>0.310</td>
<td>0.700</td>
<td>0.930</td>
<td>1.000</td>
<td>0.850</td>
<td>0.500</td>
<td>0.900</td>
<td><strong>0.734</strong></td>
</tr>
<tr>
<td>S-CR ↑</td>
<td>LC</td>
<td>0.756</td>
<td>0.923</td>
<td>0.560</td>
<td>0.919</td>
<td>0.833</td>
<td>0.870</td>
<td>0.661</td>
<td>0.793</td>
<td>0.789</td>
</tr>
<tr>
<td></td>
<td>AS</td>
<td>0.794</td>
<td>0.595</td>
<td>0.650</td>
<td>0.900</td>
<td>0.833</td>
<td>0.930</td>
<td>0.792</td>
<td>0.797</td>
<td>0.787</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>0.967</td>
<td>0.684</td>
<td>0.322</td>
<td>0.900</td>
<td>0.932</td>
<td>0.870</td>
<td>0.711</td>
<td>0.797</td>
<td>0.733</td>
</tr>
<tr>
<td></td>
<td>AT</td>
<td>0.847</td>
<td>0.485</td>
<td>0.697</td>
<td>0.930</td>
<td>1.000</td>
<td>0.966</td>
<td>0.562</td>
<td>1.000</td>
<td><strong>0.811</strong></td>
</tr>
<tr>
<td>OS ↓</td>
<td>LC</td>
<td>0.765</td>
<td>0.825</td>
<td>0.613</td>
<td>0.451</td>
<td>0.755</td>
<td>0.632</td>
<td>0.630</td>
<td>0.646</td>
<td>0.665</td>
</tr>
<tr>
<td></td>
<td>AS</td>
<td>0.654</td>
<td>0.718</td>
<td>0.577</td>
<td>0.465</td>
<td>0.659</td>
<td>0.544</td>
<td>0.599</td>
<td>0.606</td>
<td>0.603</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>0.629</td>
<td>0.577</td>
<td>0.738</td>
<td>0.464</td>
<td>0.569</td>
<td>0.571</td>
<td>0.520</td>
<td>0.522</td>
<td>0.574</td>
</tr>
<tr>
<td></td>
<td>AT</td>
<td>0.600</td>
<td>0.737</td>
<td>0.557</td>
<td>0.455</td>
<td>0.460</td>
<td>0.526</td>
<td>0.607</td>
<td>0.423</td>
<td>0.546</td>
</tr>
<tr>
<td>S-OS ↓</td>
<td>LC</td>
<td>0.565</td>
<td>0.461</td>
<td>0.613</td>
<td>0.451</td>
<td>0.533</td>
<td>0.518</td>
<td>0.528</td>
<td>0.476</td>
<td>0.518</td>
</tr>
<tr>
<td></td>
<td>AS</td>
<td>0.548</td>
<td>0.600</td>
<td>0.577</td>
<td>0.465</td>
<td>0.535</td>
<td>0.492</td>
<td>0.451</td>
<td>0.480</td>
<td>0.518</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>0.465</td>
<td>0.550</td>
<td>0.738</td>
<td>0.464</td>
<td>0.483</td>
<td>0.519</td>
<td>0.496</td>
<td>0.473</td>
<td>0.524</td>
</tr>
<tr>
<td></td>
<td>AT</td>
<td>0.523</td>
<td>0.654</td>
<td>0.558</td>
<td>0.455</td>
<td>0.460</td>
<td>0.471</td>
<td>0.574</td>
<td>0.372</td>
<td><strong>0.508</strong></td>
</tr>
<tr>
<td>SR ↑</td>
<td>LC</td>
<td>0.410</td>
<td>0.130</td>
<td>1.000</td>
<td>0.990</td>
<td>0.420</td>
<td>0.690</td>
<td>0.590</td>
<td>0.580</td>
<td>0.601</td>
</tr>
<tr>
<td></td>
<td>AS</td>
<td>0.680</td>
<td>0.420</td>
<td>1.000</td>
<td>1.000</td>
<td>0.720</td>
<td>0.860</td>
<td>0.530</td>
<td>0.640</td>
<td>0.731</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>0.600</td>
<td>0.760</td>
<td>1.000</td>
<td>1.000</td>
<td>0.822</td>
<td>0.856</td>
<td>0.922</td>
<td>0.878</td>
<td><strong>0.855</strong></td>
</tr>
<tr>
<td></td>
<td>AT</td>
<td>0.590</td>
<td>0.330</td>
<td>0.990</td>
<td>1.000</td>
<td>1.000</td>
<td>0.870</td>
<td>0.890</td>
<td>0.900</td>
<td>0.821</td>
</tr>
</tbody>
</table>

4.3 Benchmark results

We train our AD algorithms on random benign scenarios and evaluate them on SafeBench. We present the training details in Appendix A.6 and we provide important findings in the following.

Performance of AD on benign and safety-critical scenarios. The benchmark results of AD algorithms based on 4D inputs are summarized in Table 3. From the table, we observe a large performance gap of AD algorithms tested on benign and safety-critical scenarios in SafeBench. For example, although TD3 achieves an overall score of 0.830 on benign scenarios, it only achieves 0.518 when testing on safety-critical scenarios. In general, agents that perform well in benign scenarios usually fail given the safety-critical ones, indicating a trade-off between the performance under benign and safety-critical testing scenarios. For instance, PPO obtains the highest overall score on safety-critical scenarios, while its benign performance is worse than both SAC and TD3. On the other hand, although SAC achieves the highest overall score on benign testing scenarios, its performance under safety-critical ones is the worst. More results on algorithms with other types of input observations can be found in Appendix A.8.

Comprehensive diagnostic report of AD algorithms in all scenarios. In order to provide a comprehensive understanding of the performance of AD algorithms, we conducted a detailed diagnostic report for each tested algorithm from different perspectives. In particular, we consider three levels of evaluation metrics: Safety, Functionality, and Etiquette, as shown in Table 4 for the 4D-based AD agents. Comprehensive reports of all AD agents are in Appendix A.9. We observe that different AD algorithms outperform others under different metrics. For instance, on the Safety level, PPO achieves the lowest CR and OR, which means it has a high level of safety and a low accident rate, while its performance on the Etiquette level is relatively low. On the Functionality level, TD3 achieves the highest route following stability, demonstrating its ability to complete given tasks without deviating from the route. On the Etiquette level, SAC and DDPG achieve the lowest ACC and YV respectively.
With our modularized design, SafeBench is able to flexibly identify and analyze the weakness in AD systems. We place the AD agent in a fixed scene and put different AD systems. We provide a toolkit that performs physical semantic attacks in which measure the driving quality. Based on the overall score (OS), PPO is shown to be the best AD algorithm given the weighted average over all metrics.

We also notice a trade-off between functionality level metrics and safety level metrics. From Table 4, we can observe that an agent with strong functionality performance may not be safe regarding the safety level metrics. For instance, the SAC agent achieves the best TS score, which means that it can finish the routes in the shortest time, but its collision rate (CR) is also the highest among all the other agents. Similarly, the PPO agent that achieves the best route completion (Comp) score presents, however, the highest RR and SS scores, which means that it may run red lights and stop signs most frequently. This observation suggests the inherent contradiction between some safety metrics and functionality metrics, which is also unveiled in some previous studies.

### 4.4 Robustness Toolkit: physical semantic attack against AD algorithms

With our modularized design, SafeBench is able to flexibly identify and analyze the weakness in AD systems. We provide a toolkit that performs physical semantic attacks in 2 different downstream tasks and analyzes the safety and robustness of the perception modules in AD systems. We also show more examples and visualizations in Appendix A.10.

**Point cloud segmentation** To test the performance of point cloud segmentation modules, we implement 3 types of adversarial attacks: (1) **Point Attack**: a point-wise attack method that adds small disturbance to points; (2) **Pose Attack**: a scene generation method developed by us that searches pose of vehicles in the scene; (3) **Scene Attack**: a semantically controllable generative method based on SAG [51]. For Pose attack and Scene Attack, we first generate the locations and orientations of vehicles and spawn them in Carla. Then, we use the LiDAR sensor to collect point cloud needed by the segmentation algorithms. We select 4 segmentation algorithms (PointNet++ [77], PolargSeg [13], SqueezeSegV3 [79], Cylinder3D [80]) as our victim models, all of which are pre-trained on Semantic Kitti dataset [51]. We present the results of IoU after attacks in Table 3. The results show that all 4 algorithms can be attacked and have very different performances under the attack. This demonstrates the ability of SafeBench to evaluate point cloud perception modules in AD systems.

**3D object detection** To measure agents’ ability to recognize and locate surrounding objects reliably, we attack 3D object detection modules. We place the AD agent in a fixed scene and put different...
Table 5: Results of Pointcloud Attack. We test 4 pointcloud segmentation models against 3 adversarial attack methods and report the IoU results.

<table>
<thead>
<tr>
<th>Method</th>
<th>PointNet++</th>
<th>SqueezeSeg</th>
<th>PolarSeg</th>
<th>Cylinder3D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point Attack</td>
<td>0.80 ± 0.01</td>
<td>0.82 ± 0.02</td>
<td>0.94 ± 0.01</td>
<td>0.96 ± 0.01</td>
</tr>
<tr>
<td>Pose Attack</td>
<td>0.40 ± 0.08</td>
<td>0.47 ± 0.04</td>
<td>0.89 ± 0.01</td>
<td>0.88 ± 0.02</td>
</tr>
<tr>
<td>Scene Attack</td>
<td>0.52 ± 0.12</td>
<td>0.65 ± 0.04</td>
<td>0.85 ± 0.01</td>
<td>0.86 ± 0.01</td>
</tr>
</tbody>
</table>

Table 6: Performance comparison of 3D object detection. We test 2 models with different input on both benign data and adversarial data and report the average precision (AP) of car class from 3D and bird’s eye view perspectives (new 40 recall positions metric).

<table>
<thead>
<tr>
<th>Data Source</th>
<th>Model</th>
<th>Input Data</th>
<th>3D AP (%)</th>
<th>Bird’s Eye View AP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>easy</td>
<td>moderate</td>
</tr>
<tr>
<td>Benign</td>
<td>SECOND</td>
<td>LiDAR</td>
<td>87.31</td>
<td>86.81</td>
</tr>
<tr>
<td></td>
<td>CLOCs</td>
<td>LiDAR+Img</td>
<td>76.90</td>
<td>76.50</td>
</tr>
<tr>
<td>Adversarial</td>
<td>SECOND</td>
<td>LiDAR</td>
<td>60.74</td>
<td>59.81</td>
</tr>
<tr>
<td></td>
<td>CLOCs</td>
<td>LiDAR+Img</td>
<td>61.98</td>
<td>61.98</td>
</tr>
</tbody>
</table>

objects such as vehicles, pedestrians, and other traffic objects in front of the agent to test the detection accuracy. We follow TSS [82] to perform adversarial physical semantic transformations to both camera image and LiDAR point cloud. We incorporate 4 kinds of semantic transformations to attack the perception module of AD algorithms. Specifically, we first consider changing different types of vehicles such as Tesla Model 3, Audi TT, and Nissan Patrol. Second, we perturb the color of each vehicle. We only choose 5 most common colors to test the robustness of AD algorithms although in practice any RGB value can be applied to the car in SafeBench. Third, for pedestrians, we include a diverse set with a variety of body shapes and skin colors. Finally, we perform rotation to every object to examine the reliability of AD systems. We present our results in Table 6. We train 2 different 3D object detection models on normal driving scenarios and test the models on adversarial data generated by SafeBench. The SECOND [83] takes LiDAR point clouds as input while CLOCs [84] is a multi-modal model that takes both LiDAR point clouds and camera images. From the table, we find that the SECOND model performs better on benign data. However, on adversarial data, CLOCs achieves higher average precision and the performance drop of the CLOCs model from benign to adversarial is much smaller than that of the SECOND model. One of the reasons is that data from both modalities complement each other, helping the model to make better decisions, which indicates potential designing strategies for AD algorithms: use multi-sensor fusion module to incorporate and process multi-modal data, leading to higher robustness. This demonstrates the ability of SafeBench to evaluate object detection modules.

5 Conclusion

In this paper, we introduce SafeBench, the first unified platform to automatically evaluate and analyze the performance of AD algorithms in multiple aspects using various safety-critical driving scenarios generated by different generation algorithms. We incorporate 8 safety-critical scenarios and 10 evaluation metrics from 3 different levels to provide a detailed diagnostic report for each AD agent. AD algorithms tested on SafeBench have a large performance drop compared to evaluations on benign scenarios, suggesting the deficiencies of each algorithm and the effectiveness of our testing platform. We hope our platform and findings will serve as a reliable and comprehensive benchmark to help researchers and practitioners to identify weaknesses in existing AD systems and further develop safe AD algorithms as well as more effective testing scenario generation algorithms.

Limitations Although simulation is a useful and necessary tool for evaluating AD systems given its efficiency and controllability [85, 86], the simulation in SafeBench cannot exactly reflect real-world conditions. On track testing is necessary before deploying AD algorithms in the real world. Besides, we only evaluate RL based AD algorithms in the current version of SafeBench, and testing more diverse AD algorithms, including commercial systems such as Baidu Apollo [87] and Openpilot [88], would be interesting future work.


[55] Stanford Artificial Intelligence Laboratory et al. Robotic operating system.


Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes] See Section 1.
   (b) Did you describe the limitations of your work? [Yes] See Appendix A.7
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix A.3
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See in the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix A.6
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Appendix A.6
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix A.6

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 3.3
   (b) Did you mention the license of the assets? [Yes] See in the supplemental material.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See in the supplemental material.
   (d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See Appendix A.7

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]