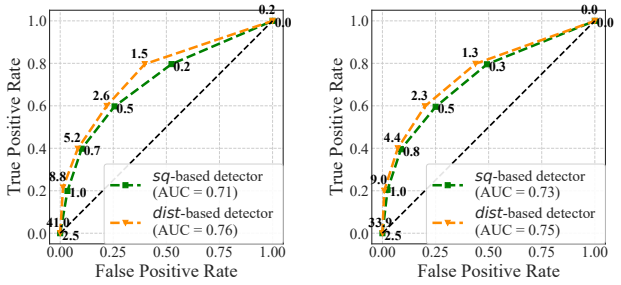
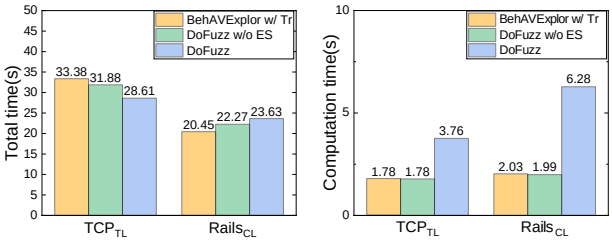


(a) $Dataset_1$ (b) $Dataset_2$

Figure 16: Trajectory geometry relationship *sq* and trajectory distance *dist* based anomaly detection for type-wrong approximated scenarios.



(a) Total Time (b) Computation Time

Figure 15: Real-world time cost in the test process, for each scenario on average. Computation time is total time minus simulation time.

A PARAMETER SETTINGS OF ADS AND BASELINE FUZZER

For targeted ADS, as stated in Section 4.5, we utilize publicly available code and pre-trained models to avoid incorrect implementation, consistent with past practices in testing ADS [21]. The purpose of testing is to identify potential flaws in the development stages of ADS, including incomplete training data, insufficient training processes, or logical flaws. Therefore, to ensure fairness in ADS testing, it is advisable to use out-of-the-box ADS. Additionally, in this work, we validate the driving capabilities of each ADS in blank control scenarios (removing all NPCs) to ensure that the discovered ADS flaws are triggered by manipulated test scenarios.

For baselines, the primary difference between deployment on CARLA and other simulators lies in the interface between the testing algorithms and the simulator, specifically in the manipulation of NPCs. Following existing practices [43, 44], when transplanting these baselines to the CARLA environment, we standardized the way they manipulate NPCs as mutating NPCs' route and its corresponding waypoints [8]. Additionally, due to CARLA API not directly supporting trajectory input, we implemented a PID-based trajectory following component (mentioned in Section 4.5) for these baselines.

B FUZZING OVERHEAD

To assess the impact of DoFuzz's effectiveness on computation time (total time minus simulation time), we compare the real-world time cost of AV-Fuzzer, DoFuzz w/o ES, and DoFuzz in Fig. 15. The computation time of DoFuzz is more than twice as long as both baselines, as it involves intensive computation to estimate the Exception Score of each test input. However, it is obvious that the additional computation cost is still much slighter than the most time-consuming part, i.e., the simulation cost. Asynchronous techniques for simulation and computation can save computation time, which is out of our scope in this paper. Interestingly, as shown in Fig. 15, the additional total cost is less than the additional computation cost, and in TCP_{TL} setting, the slight computation time addition (2s) even brings back reduction in average simulation time by more (6.7s). It occurs because DoFuzz successfully deprioritizes adequately-exploited immobile test inputs, like cases in Fig. 10 (a) (b) (d) (e) (f), postponing or canceling such long-execution-step trips within the fuzzing budget. Note that while the average time cost per scenario may increase with our approach (about 15% in Rails_{CL} setting), the efficiency improvement (about 50% in Rails_{CL} setting) in discovering unique failure scenarios brings higher benefits.

C FEASIBILITY OF *EXCEPTION* ESTIMATION

The high-level idea of our *exception* estimation is to measure the possibility that the test input mismatches any clusters of prior tested scenarios. We implement it with an interpretable methodology: **First**, we transform the test input s into the scenario representation Sce'. This process depends on the retrieved knowledge from prior tested scenarios (e.g., trajectory of *Ego vehicle*). For this step, i.e., Scenario Approximation (SA) described in §3.3.2, we expect to retrieve the best knowledge as long as it exists, so that we can obtain an accurate Sce' relative to the actual execution result Sce. In other words, for each s, if the Frequency of Prior type-Identical test inputs (FPI) > 0, **we expect** the deviation between Sce' and Sce close to 0, and vice versa. **Second**, we assign Sce' a plausibility score. **We expect** low plausibility indicates that wrong knowledge is retrieved for the test input s in step SA.

Fig. 12 presents the deviation (SAd) between Sce' and Sce against FPI and ES as well. Basically, the lower the Frequency of Prior type-Identical test inputs (FPI), the less precisely SA approximates, and the higher the ES. This means not only the Sce' is generated as we expect, but also the plausibility-based ES is calculated as we expect. Additionally, we create two ES thresholding detectors to identify the type-wrong knowledge retrieval events in step SA. Fig. 16 demonstrates that the detectors (based on both the plausibility metric proposed in §3.3.2 and a *dist* metric that is the minimum trajectory distance between Sce' and existing clusters) achieve high ROC-AUC scores. This confirms our plausibility metrics can determine the type-wrong knowledge retrieval events effectively. Note that while the trajectory geometry-based plausibility contributes to slightly lower ROC-AUC scores than the trajectory distance-based one, its threshold values remain more stable across different ADS fuzzing datasets, making it potentially more promising for future ADS scenario studies.