A Differential Testing Framework to Identify Critical AV Failures Leveraging Arbitrary Inputs

Trey Woodlief*
University of Virginia
Charlottesville, VA, USA
adw8dm@virginia.edu

Carl Hildebrandt*
University of Virginia
Charlottesville, VA, USA
hildebrandt.carl@virginia.edu

Sebastian Elbaum
University of Virginia
Charlottesville, VA, USA
selbaum@virginia.edu

***Abstract*—The proliferation of autonomous vehicles (AVs) has made their failures increasingly evident. Testing efforts aimed at identifying the inputs leading to those failures are challenged by the input's long-tail distribution, whose area under the curve is dominated by rare scenarios. We hypothesize that leveraging emerging open-access datasets can accelerate the exploration of long-tail inputs. Having access to diverse inputs, however, is not sufficient to expose failures; an effective test also requires an oracle to distinguish between correct and incorrect behaviors. Current datasets lack such oracles and developing them is notoriously difficult. In response, we propose DIFFTEST4AV, a differential testing framework designed to address the unique challenges of testing AV systems: 1) for any given input, many outputs may be considered acceptable, 2) the long tail contains an insurmountable number of inputs to explore, and 3) the AV's continuous execution loop requires failures to persist in order to affect the system. DIFFTEST4AV integrates statistical analysis to identify meaningful behavioral variations, judges their importance in terms of the severity of these differences, and incorporates sequential analysis to detect persistent errors indicative of potential system-level failures. Our study on 5 versions of the commercially-available, road-deployed comma.ai OpenPilot system, using 3 available image datasets, demonstrates the capabilities of the framework to detect high-severity, high-confidence, long-running test failures.**

***Index Terms*—differential testing, autonomous system validation, autonomous vehicles**

I. INTRODUCTION

There are many autonomous vehicles (AVs) on the roads today, and their increased presence is making their failures and the consequences of those failures more common [1]–[5]. A critical issue in testing AVs is their long-tail distribution of inputs. This includes a multitude of rare and unusual scenarios, which are not frequently encountered during operation but can lead to unexpected behaviors and sometimes catastrophic failures. The number and variety of these edge cases underscore the importance of comprehensive and rigorous testing methodologies for AVs to ensure their safety and reliability.

We hypothesize that leveraging this massive, continuously growing volume of sensor data, alongside open access to existing diverse datasets, provides a unique opportunity to uncover and analyze the long tail of sensor inputs—especially those edge cases not previously encountered. This volume of sensor data being collected by AVs is increasing rapidly.

Fig. 1: DIFFTEST4AV identifies this input image as one causing a high-impact failure: over 90% confident the SUT response to that image is an outlier when compared with the reference systems, and causes a 20° output difference. SUT steering (red line) causes the vehicle to turn off-road.

For instance, the California Department of Motor Vehicles reported that registered AVs logged 2 million miles in 2020 [6], 4 million miles in 2021 [7], 5.7 million miles in 2022 [8], and over 9 million miles in 2023 [9]. Effectively mining these datasets can reveal deficiencies in newer system versions, which might otherwise go undetected.

Having more and diverse inputs, however, is necessary but not sufficient to build a test that exposes failures. A test consists not just of an input but also of an oracle that can distinguish between correct and incorrect behavior of that input on the system under test (SUT). Existing sensor input datasets for AVs tend to lack an oracle, and developing oracles for them is notoriously challenging [10]. At best, these datasets provide pieces of an oracle, such as the output of a human driver or the output of another system, but they often lack sufficient context and provenance details. For instance, while a human driver's actions can offer some insights, they constitute just one of many possible behaviors, and they do not account for the person's state or driving style. This one action may be just one correct answer out of many; treating it as the sole answer would be too narrow of an oracle. Further, this action may be incorrect or an outlier, coming from a distracted driver. In either case, this is insufficient to serve as an oracle directly.

To address the AV oracle challenge, we build on prior work on *differential testing*, which has sought to address the problem

*Equal Contribution

of lacking oracles for traditional software by using other systems to create pseudo-oracles. Given the SUT and a reference system S aiming to satisfy the same specifications, differential testing oracles check that $\|SUT(input) - S(input)\| < \delta$ for a suitably small δ. In the case of AVs, the common practice of frequent releases (e.g., OpenPilot's software is contributed to hourly [11] and Tesla's self-driving software is updated every few weeks [12]) offers a common path towards identifying viable systems similar to SUT to perform differential testing.

However, there are key differences between AV systems and traditional software systems that limit the direct application of differential testing to AV systems. First, for any given input scenario, **a range of output behaviors might be acceptable** [10]. In other words, the acceptance threshold δ between behaviors depends on the particular inputs. For some inputs, small variations are expected. For example, in the presence of congested standstill traffic, the acceleration and steering should always be close to 0. Other inputs may have behavior that varies slightly; for instance, some AVs might merge more assertively, resulting in a quick, sharp steering angle, while others may merge slowly, resulting in a long, extremely small steering angle. There are also cases where large variations in output across AVs may be acceptable. For example, to avoid a collision, a vehicle may steer aggressively, brake, or both. To address this variability, an AV differential framework must not only identify when an output differs across systems but also provide a reliability or confidence estimate that the difference is not within the range of acceptable answers δ.

Second, given the input dataset sizes for AVs, there will be **an insurmountable number of tests** revealing, with high confidence, that there is a difference between the SUT and S. An AV differential framework must be able to not only identify these tests, but also provide an estimate of the severity of failing tests. For example, being confident that we can reliably produce a steering angle difference of 1° in a system is probably much less severe than a test of equal confidence that produces a steering angle difference of 20°. We refer to a high-confidence, high-severity failure as *high-impact*. Figure 1 highlights a high-impact failure found by our approach where the SUT attempts to turn right at a curve to the left. This failure has a confidence of 99.3% and a severity of 22.2°.

Third, AV systems operate in continuous time, constantly receiving new inputs and producing output commands to actuate in the world in real time. Even if the AV's output is deemed incorrect at any specific instant, this constant loop **attenuates the ability for a single failure to propagate to a system-level failure**. For example, a steering angle output that is 10° off for a few milliseconds during one prediction cycle but then gets corrected in the next may lead to no perceptible change since the system could not turn 10° in that time. Meanwhile, a steady error of 10° over several seconds could pull the AV into the opposing lane of traffic. Accordingly, an AV differential framework must address this temporal aspect in its analysis, automatically identifying long-running failures that are more likely indicative of potential system-level failures. By extending the definition of differential testing, this framework must not only detect differences at discrete moments but also evaluate the persistence of these differences over time to ensure comprehensive system reliability.

To address these requirements, we propose DIFFTEST4AV, a differential testing approach for AVs and other autonomous systems that accounts for their unique operational paradigm, enabling developers to use arbitrary inputs to test AV systems. DIFFTEST4AV identifies high-severity, high-confidence, long-running failures that could lead to system-level failures. To do so, it first uses statistical analysis to contrast the outputs between the SUT and reference systems to find high-confidence differences. Second, it evaluates the severity of the differences to prioritize critical issues. Third, it analyzes the confidence and severity over time to detect persistent software failures indicative of potential system-level failures. We evaluated DIFFTEST4AV's ability to uncover failures on the commercially-available, road-deployed comma.ai OpenPilot system from three real-world datasets. Our findings indicate that **DIFFTEST4AV can identify high-confidence, high-severity, long-running failures for state-of-the-art systems on arbitrary sensor data**, identifying 143 inputs out of 4.58 million (0.003%) that yield high-impact failures at over 90% confidence and 40° severity, including a 27-input (1.8 second) duration failure—a failure of this duration and severity would result in potentially catastrophic failures for the AV.

We begin with a brief overview of the relevant background and related work in Section II. Then, in Section III we describe the DIFFTEST4AV framework in detail, discussing the overall problem statement and providing motivation for and solutions toward the three identified requirements of confidence, severity, and duration of failures. Finally, in the experiment in Section IV, we demonstrate the utility of DIFFTEST4AV through an imagined continuous integration testing paradigm for the commercial, road-deployed comma.ai OpenPilot system to identify high-impact, long-running failures.

II. BACKGROUND AND RELATED WORK

A test $oracle$ is a function that distinguishes between correct and incorrect behaviors of an SUT [10], [13]. An oracle takes an input and the system's output, and maps them to a Boolean value, $oracle : (input, SUT(input)) \mapsto \mathbb{B}$. Automating the oracle function is key to scaling up the testing process to the size of modern AV datasets. For an oracle to automatically perform this operation, it typically compares the observed output with a known expected output. The methods of deriving the expected output and performing the comparison are implicit oracles, specification-based oracles, and differential oracles.

Implicit test oracles rely on the premise that there is agreement among stakeholders that some post-conditions are unacceptable. For example, automated fuzzing tools [14]–[16] often assume that a segmentation fault or uncaught exception is incorrect. Such implicit oracles are effective at detecting the most extreme misbehaviors and tend to be easy and inexpensive to check, but can only map very few inputs.

Specification-based oracles generalize to more inputs and subtle categorization of behaviors as per the expected behavior

based on the specification [13], [17]–[20]; however, this requires oracles be developed for each specification evaluated.

Differential oracles utilize reference systems with the same specification as the SUT to judge correctness, eliminating the need to craft individual oracles per specification [21]. Given an input, if the reference system and the SUT generate the same behavior, then they are deemed consistent; otherwise, if the reference system is deemed correct, then any inconsistency is a failure. While this retains the strength in failure identification of specification-based oracles, an individual failure can no longer be linked to a particular specification.

We now discuss related work in these dimensions. Implicit oracles for AVs have focused on clearly incorrect behaviors, such as failing to arrive at the destination [22], driving off the road [23], driving in an opposing lane [24], or causing collisions [25]–[30]. While lightweight, these are limited in the depth of failures identified.

In specification-based oracles for AVs, most efforts check for postconditions on systems' output state (e.g., maximum velocity, minimum battery, waypoints within a reachability range, or aligning with traffic rules) [10], [31]–[34] that apply to all inputs. Developing more general specifications that relate the system sensor inputs and system state to system behavior is much more challenging, in part due to the size and complexity of the input space and the range of acceptable behaviors. To address this challenge, metamorphic specifications have emerged to guide the oracle function in AVs. Such specifications provide auxiliary functions describing how changes in inputs relate to changes in outputs [35], [36]. For example, there have been a myriad of approaches to apply transformations to camera images that mimic sensor noise, weather, or lighting changes that should not change the AVs steering or acceleration by more than a threshold [37]–[41], or changes to particular portions of the image that should change the system behavior in specific ways (e.g., adding a vehicle in front of ego should cause ego to deaccelerate, changing a light to green should not decrease the ego's velocity) [28], [42], [43]. Attempts to increase the power of metamorphic functions, however, are limited as jointly changing streams of inputs from multiple sensors and estimating the effect on the system output is approximating the difficulty of developing the system itself. The continued complexity in building specification-based oracles demonstrates the potential benefit of differential testing to lower the overhead of AV testing.

Differential testing has been applied to autonomous systems, from comparing aviation software operations [44] to comparing AV behaviors with human drivers [45]. Unlike prior work, DIFFTEST4AV does not require a ground truth output for comparison. Instead, it introduces a novel differential testing approach that accounts for the unique operational paradigms of AV systems, enabling the identification of inconsistencies without relying on predefined "correct" outputs.

III. APPROACH

DIFFTEST4AV aims to identify test inputs among vast field datasets that induce high-impact failures in an AV system. It is a differential testing approach for AV and other autonomous systems that accounts for their unique operation paradigm, enabling developers to use arbitrary inputs to test AV systems, identifying high-severity, high-confidence, long-running software failures that could lead to system-level failures.

A. Problem Definition

An autonomous vehicle, denoted as AV, navigates scenarios using a combination of its sensor readings of the environment, denoted as x, and its current internal system state, represented by s. Sensor readings x include data from cameras, LiDAR, radar, and other sensors that help perceive the environment. Examples of the system state s include the vehicle's velocity, position, acceleration, and current steering angle. These inputs are processed by the AV, which computes the appropriate action a, denoted as $a = AV(x, s)$. The chosen action a alters the AV's state in the world and subsequently affects the vehicle's future sensor readings. Note that sensor values and states are matched pairs, i.e. s_i is the system state during which x_i was observed; let $t_i = (x_i, s_i)$ refer to these matched inputs; for brevity we say $a = AV(t_i)$. Recall that a test is comprised of an *input* and an *oracle*; here t_i constitutes the test input, and DIFFTEST4AV will derive an oracle that takes in t_i and $AV(t_i)$ and decides whether the AV passed or failed.

AVs operate continuously in the world, constantly observing inputs and producing actions, creating a continuous feedback loop of action and reaction. Over time, as the AV operates within a scenario, it observes a sequence of sensor and state pairs $\vec{T} = \langle (x_0, s_0), \dots \rangle$; let $\vec{A} = AV(\vec{T})$ refer to the corresponding sequence of AV actions. We use array index notation to refer to sequence elements, e.g., $\vec{A}[j] = a_j$ and $\vec{A}[j:k] = \langle a_i \in A \mid j \leq i \leq k \rangle$. We now generalize the notion of oracle to consider sequences of inputs and outputs; in the AV domain we are particularly interested in identifying long-running software failures as they are more likely to lead to system-level failures. Formally, given a test $\vec{T}$, DIFFTEST4AV aims to identify the set of failing subsequences $\vec{F} \in FailSet$:

$$FailSet = \{\vec{F} \mid \vec{F} = \vec{T}[j:j+m] \subseteq \vec{T} \wedge \neg oracle(\vec{F}, AV(\vec{F}))\}$$

We can generalize these definitions to multiple tests which is important as the AVs operating globally today produce massive amounts of arbitrary observed sensor and state data, denoted as $TestSet = \{\vec{T_0}, \dots\}$. Testers can access portions of this data from their own datasets, by combining datasets, or by using publicly available external data collected by an other AV's or sources such as dashcams. Independent of the source, DIFFTEST4AV provides the mechanism to identify failing subsequences from arbitrary test inputs. For the rest of this section, we will present the analysis for a single-test case, referring to a test $\vec{T}$. However, any analysis with multiple tests would simply repeat the process for each separate test.

B. Differential Testing

We propose using differential testing, which leverages other systems built for the same interface and specifications that

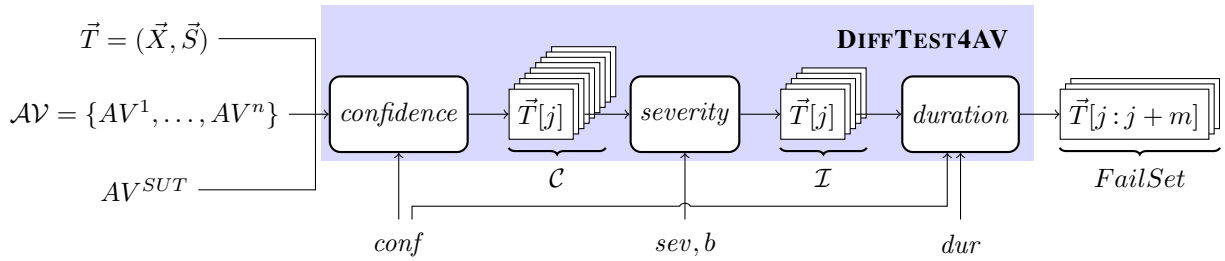


Fig. 2: DIFFTEST4AV pipeline for a single test case $\vec{T}$.

can serve as reference systems to identify failures in the SUT, AV^{SUT}. Given a set of n autonomous systems $\mathcal{AV} = \{AV^1, AV^2, \ldots, AV^n\}$ that share the same interface and specifications, the approach runs each vehicle on the input to obtain their actions. DIFFTEST4AV then compares the actions taken by the reference systems to that of the SUT and derives a notion of confidence and severity of failure for the SUT.

Figure 2 illustrates the components of DIFFTEST4AV. In the first stage, DIFFTEST4AV begins by identifying the inputs in $\vec{T}$ that cause high-confidence failures, where the level of confidence is decided by the user through the parameter $conf$. In the second stage, these high-confidence failure inputs are filtered to preserve only those that also yield high-severity failures, where the threshold of severity is parameterized by sev. We refer to high-confidence, high-severity failures as *high-impact* failures. Finally, the high-impact failures are used to identify long-running failures over a given duration. This step is parameterized by both the threshold for long-running, dur, and the confidence threshold. In the following sections we elaborate each of these components and explore a range of parameter choices for $conf$, sev, and dur.

C. Confidence: Statistical Methods for Failure Identification

Driving this component is the insight that the problem of identifying when AV^{SUT} fails with respect to the n other AVs can be seen as an instance of the outlier detection problem in statistics [46]. Given a sample and a data distribution, outlier detection aims to determine the likelihood that the sample did not come from the distribution, i.e. what is the probability it is an outlier. This component aims to answer a similar question. Formally, given a test input $t_i \in \vec{T}$, a set of outputs $ActionSet = \{AV(t_i) \mid AV \in \mathcal{AV} \cup AV^{SUT}\}$, what is the probability that $AV^{SUT}(t_i)$ is an outlier in $ActionSet$?

A difficulty is that statistical approaches require a characterization of the underlying data distribution to reason about outlier probabilities. However, completely characterizing the distribution of acceptable outputs in response to a certain input is impracticable. In the absence of a known data distribution, the sampled data points, e.g. $ActionSet$, can be used to estimate the distribution and identify outliers. The appropriate choice of statistical estimation method depends on several conditions including the data being analyzed, assumptions about its distribution, the power requirements, and the number of data points being analyzed, e.g. the number of prior available systems. For example, under the assumption of normally-distributed data, Dixon's Q Test [47], the one we later used in our study, is suitable for a relatively low number of available systems, e.g. 2-5 systems in addition to the AV^{SUT}, while Grubbs' Test [48] is better suited when there are more systems. Other outlier detection methods may be appropriate in other situations [49]; our approach is configurable to utilize different statistical approaches as appropriate. This produces a function $confidence(t_i, AV^{SUT}, \mathcal{AV})$ that outputs, from 0% to 100%, the confidence that the output of AV^{SUT} on t_i is an outlier based on the other systems $\mathcal{AV}$ (Section III-E later shows how we extend this to sequences of inputs).

Using this confidence estimation function, DIFFTEST4AV can now address the first threshold to find all inputs that yield a potential failure over the confidence threshold $conf$. Let $\mathcal{C}$ be the set of high-confidence failing inputs:

$$\mathcal{C} = \{\vec{T}[j] \mid confidence(\vec{T}[j], AV^{SUT}, \mathcal{AV}) \geq conf\}$$

Example. In Section IV we explore a parameterization of DIFFTEST4AV using Dixon's Q Test [47] for the $confidence$ function. Under the assumption of normally distributed data[1], Dixon's Q test uses the ratio between the gap, the distance between the most extreme data point and its next nearest data point, and the $range$, the distance between the largest and smallest data point, to derive a probability that the most extreme data point is an outlier, i.e. the confidence[2]. This is the namesake statistic, $Q = {}^{gap}/_{range}$. The Dixon's Q Test sidesteps the issue of needing to directly compute the parameters of the underlying distribution through the use of ratios, which makes it particularly well-suited to use-cases where there are limited numbers of samples to support such calculations. The confidence is based on both the test statistic, Q, and the number of samples tested. For our approach, the number of samples tested is equal to the number of reference systems plus one, $|\mathcal{AV}| + 1$, as the SUT is included in the number of samples. The more reference systems there are, the lower the Q value required for the same confidence. To achieve 90% confidence for 3 samples requires $Q \geq 0.89$, 4 requires $Q \geq 0.68$, and 5 requires $Q \geq 0.56$ [47]. Concretely,

[1]Dixon's Q test is robust to non-normal data for small sample sizes [50], though further research is needed to define the boundaries of application.

[2]Note that if the SUT is not the most extreme data point, judged by distance from the sample mean, then we conservatively set the confidence to 0 as this indicates we are more confident that a reference system is failing instead.

Fig. 3: $> 98\%$ confidence failure, frame 404 in Table I.

Fig. 4: $> 91\%$ conf., 0.16° sev., frame 222 in Table II.

TABLE I: SUT Failures over Several Frames

Frame	402	403	404	405	406
	Steering Angle (degrees)				
AV^1	3.90	3.90	2.57	1.25	1.25
AV^2	7.29	6.36	5.25	4.06	4.06
AV^3	0.10	0.10	0.08	0.08	0.06
AV^4	11.40	7.67	7.67	8.93	8.93
AV^{SUT}	31.23	33.37	33.37	33.37	20.35
Gap	19.83	25.70	25.70	24.44	11.42
Range	31.13	33.27	33.29	33.29	20.28
Q	0.637	0.773	0.772	0.734	0.563
Confidence	94.76%	98.87%	98.86%	98.12%	90.35%

TABLE II: High Confidence Low Severity SUT Failure

Frame	220	221	222	223	224
	Steering Angle (degrees)				
S_1	0.34	0.34	-0.07	-0.19	-0.30
S_2	-0.12	-0.13	0.00	-0.40	-0.47
S_3	-0.02	-0.02	-0.02	-0.02	-0.02
S_4	0.12	0.12	-0.11	-0.11	-0.64
AV^{SUT}	-0.05	-0.05	-0.27	-0.27	-0.27
Gap	—	—	0.16	—	—
Range	—	—	0.27	—	—
Q	—	—	0.772	—	—
Confidence	0%	0%	91.33%	0%	0%

if for a given input the 4 reference systems' outputs lie between 0 and 43, and the SUT output is 100, then this gives $range = 100 - 0 = 100, gap = 100 - 43 = 57, Q = {}^{57}/_{100} = 0.57 > 0.56$, so we are more than 90% confident the SUT is an outlier, i.e. this input causes AV^{SUT} to fail.

Consider the example sequence of outputs for $\mathcal{AV}$ and AV^{SUT} shown in Table I and Figure 3. All examples are taken from the open-source real-world OpenPilot 2k19 dataset applied to 5 commercial AV systems explored and further discussed in Section IV. The image is a dashcam input for a steering control system. The outputs shown in Table I are the steering angle that each system predicts the AV should actuate based on the given input. The different columns correspond to the consecutive input frames immediately before and after the input shown in Figure 3. From the table, we see that the outputs produced by the reference systems in frame 404 are in relative agreement, predicting a steering angle from 2 to 8 degrees, i.e., steer slightly left. By contrast, the SUT outputs a much stronger signal to turn left by 33 degrees. While the image appears to show the AV on the right edge of the lane, and thus a correction to the left may be warranted, the output of the SUT may be closer to turning left than centering in the lane. The Dixon's Q test to compute the gap, range, Q, and confidence, shown in the lower half of Table I. Here we find, with over 90% confidence in each frame, that the SUT failed.

D. Severity: Identifying SUT Misbehavior

In addition to identifying failures with high confidence, we also aim to identify failures with high severity. More precisely, given a single test input t_i, $severity(t_i, AV^{SUT}, \mathcal{AV})$ rates the severity of a failure of AV^{SUT} based on the outputs of the reference systems relative to the test input. The simplest approach is to measure the difference between the outputs, e.g. to use the difference in steering angle as the severity. However, we can further enrich this by using the test input to estimate how the output will affect the system's behavior. Let b be a function that takes as input the test input and the system output, and produces an estimate of the system's future behavior. The behavior function can use the system state, output action, and kinematics to, e.g., estimate the future position of the SUT and the reference systems. Analyzing the behavior will allow for, e.g., distinguishing that a steering error of 5° is potentially catastrophic at highway speeds, but will have no effect if the vehicle is parked. The *severity* function then finds the minimum distance between the SUT's behavior and that of any of the reference systems:

$$severity(t_i, AV^{SUT}, \mathcal{AV}) = \min_{AV \in \mathcal{AV}} \|b(t_i, AV(t_i)) - b(t_i, AV^{SUT}(t_i))\|$$

Using this function, DIFFTEST4AV can find all test inputs that yield a potential failure over a given severity threshold sev: Let $\mathcal{I}$ be the set of high-impact failing inputs:

$$\mathcal{I} = \{\vec{T}[j] \mid \vec{T}[j] \in \mathcal{C} \wedge severity(\vec{T}[j], AV^{SUT}, \mathcal{AV}) \geq sev\}$$

Example - the need for severity. Figure 4 and Table II are presented in the same format as the previous example. Note that values of "—" represent cases where the SUT output was within the range of the reference system outputs, so it is not a candidate for a failure. In this case, as will be further explored

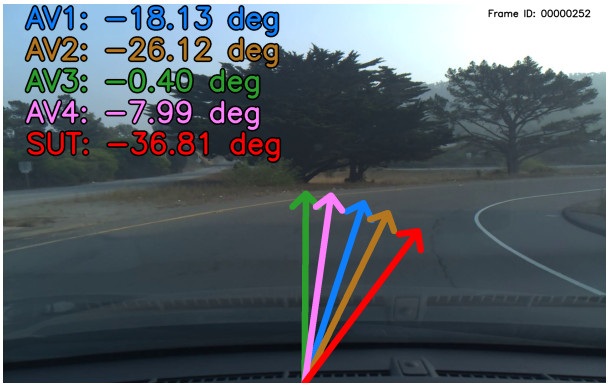


Fig. 5: 57.6% confidence, 10.7° severity failure.

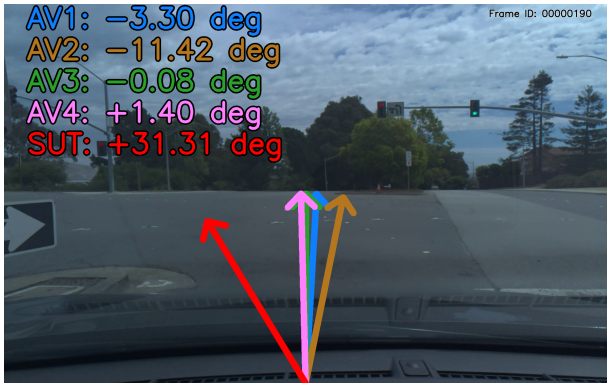


Fig. 6: Image with $> 97\%$ confidence failure, out of ODD

in Section IV, the behavior function b is given by the identity function, e.g. the system behavior is approximated through the steering angle output. In frame 222, shown in the middle column and depicted in the image, the SUT outputs a value of -0.27° which is more than double the next closest value, while the rest of the outputs are closely clustered. Using Dixon's Q test results in a value of $Q = 0.772$ which corresponds to a confidence of 91.33%. However, the severity, as indicated by the gap value in Table II, is only 0.16°. Although this is large in relative terms as compared to the other outputs, leading Dixon's Q test to yield high confidence, this is minuscule in terms of steering output, leading to a low severity. As we can see from the full array of steering angles in Figure 4, the outputs are oscillating near 0 for all systems across all frames. Thus, although true 0 may be the "optimal" output, all systems are likely demonstrating acceptable behavior.

In contrast, a severity threshold of 20° would identify the failure in Figure 3 while ignoring the high-confidence outlier in Figure 4. This showcases the utility in using a severity threshold when identifying failures.

Example - when severity is not enough. In contrast to the previous example which showcased the problems that arise from high-confidence but low-severity failures, let us now examine a case of low confidence but high severity. Figure 5 demonstrates an input that yields a high-severity (10.7°), but low-confidence (57.6%) failure. The low confidence is due to the large spread of outputs from the reference systems with $Q = {}^{gap}/_{range} = {}^{10.7}/_{36.4} = 0.29$. This spread and low confidence indicates that this should not be identified as a failure of the SUT. However, we note that a different use case for this approach could identify high-severity, low-confidence failures as these likely point to a failure in one or more of the reference systems as well due to the large proportion of disagreement; we leave such investigation for future work.

E. Duration: Identifying Potential System-Level Failures

Although the prior two sections allow DIFFTEST4AV to identify the severity and confidence of failures in the SUT based on single-instant sensor inputs, we are particularly interested in identifying test-input sequences that can lead to prolonged failures as these may escalate to system level failures. One simple method for doing so is to identify consecutive failures by the SUT that meet the sev and $conf$ thresholds for a given duration dur. Let $FailSet_{simple}$ be the set of high-impact failures of at least duration dur:

$$FailSet_{simple} = \{\vec{T}[j:j+m] \mid 0 \leq j \leq j+m < |\vec{T}| \wedge m \geq dur \wedge \forall t_i \in \vec{T}[j:j+m], t_i \in \mathcal{I}\}$$

However, this does not account for the fact that the confidence of an entire sequence yielding a continuous failure is lower than the confidence that at least one of the input yields a failure. Thus, we adjust the requirement to be more strict by taking the product of the individual failure confidences to find a confidence in the entire sequence yielding a continuous failure. Let $FailSet$ be the set of high-impact failures of at least duration dur and combined confidence $conf$:

$$FailSet = \{\vec{T}[j:j+m] \mid \vec{T}[j:j+m] \in FailSet_{simple} \wedge \prod_{t_i \in \vec{T}[j:j+m]} confidence(t_i, AV^{SUT}, \mathcal{AV}) \geq conf\}$$

Note that using the product to estimate the confidence of a continuous failure requires the assumption that the likelihood of failure on successive sensor inputs of each AV system is independent. Yet in practice this is likely not the case since successive inputs are likely to be highly similar. However, even if this assumption does not hold, the failure sequences found by taking the product are a subset of the original failure sequences found, e.g. $FailSet \subseteq FailSet_{simple}$, all of which have higher average confidence than the original, which may be useful to the tester as this estimate is more conservative. Once the failing sequences have been identified, they can be additionally filtered for maximality or to identify only non-overlapping sequences based on testing goals.

Example. Let us re-examine the data from the first example shown in Figure 3 and Table I. If we use a confidence threshold of 90%, each individual frame meets the criteria; under the simple aggregation technique, all five frames would represent a maximal failure in $FailSet_{simple}$. However, if we use the more conservative technique, we find that only

certain subsequences are retained in $FailSet$. For example, frames 402 to 405 have a product of $94.76\% \times 98.87\% \times 98.86\% \times 98.12\% = 90.88\%$ which meets the threshold of 90%. However, adding frame 406, would bring the overall confidence to $90.88\% \times 90.35\% = 82.11\%$ which is below the threshold. This means that while $\vec{T}[402:405] \in FailSet$, $\vec{T}[402:406] \notin FailSet$.

Taken together, the three stages of DIFFTEST4AV can identify, from arbitrary sequences of sensor inputs, the SUT's highest-severity, highest-confidence, longest-duration failures.

F. Limitations and Extensions

We now briefly discuss the limitations of our approach and potential extensions for future work to address these.

Requiring Multiple Reference Systems. Our approach assumes that sensor data is readily available and that it is a superset of the data consumed by the autonomous systems. It also assumes the availability of multiple autonomous systems with identical specifications. These assumptions are gradually becoming less restrictive as more vehicles, and more vehicle versions, from various companies, capable of generating these sensor datasets, begin to operate on our roads.

Although the strongest form of differential testing requires multiple independent implementations, DIFFTEST4AV only requires a single execution of the test suite per version. That is, each time a new version of the system is developed the new version is treated as AV^i and exercised over the test suite, $TestSet$, and its actions recorded. Over time, this naturally builds a library of reference system outputs that can be analyzed using DIFFTEST4AV to identify failures, with the most recent version treated as AV^{SUT} and the outputs from the prior versions forming $\mathcal{AV}$. This process fits the continuous integration (CI) testing workflow already used in AV development [51] and is efficient. By storing the results from prior tests for each version, only the most recent version must be exercised each time. The study in Section IV assumes this workflow, using four previous versions of the same AV system as the reference systems to evaluate the latest version.

Statistical Assumptions. Reasoning about whether the outputs of the AV systems meet the assumptions of various statistical tests is a difficult problem. Many tests of the assumptions require large quantities of data and become more difficult to process in the face of potential outliers. Particularly, this problem setting requires reasoning about the distribution of outputs for all of the tested systems *conditioned by the input*. If the assumptions do not hold, then DIFFTEST4AV may over or under estimate the outlier probability. However, DIFFTEST4AV can still provide utility. Rather than using the confidence as a threshold with defined semantics, it can be used as a prioritization criteria. That is, instead of investigating all tests where there is 90% confidence it is an outlier, the developer should investigate tests with 99% confidence before tests with 90% confidence since the relative strength of $99\% > 90\%$ holds even if the actual probability is incorrect.

Parameterization. DIFFTEST4AV relies on thresholds for confidence, severity, and duration which must be manually tuned to suit the specific testing goals of the end user. As discussed above, these metrics can be used as a sort criteria rather than a filtering critera. However, prioritizing across the three dimensions remains on open challenge. Future work should examine methods to unify these three dimensions, perhaps creating a blended metric incorporating confidence, severity, and duration to rank the highest-priorty failures.

Handling Multiple Outputs. As presented, DIFFTEST4AV only identifies failures of systems that produce a single numerical output, e.g. an AV system that produces a steering angle. However, modern AV systems often produce many different outputs of different kinds. We believe that natural extensions to DIFFTEST4AV could allow for handling multiple outputs over numerical types. For more complex output types, additional refinement is required. For example, an AV system may output a set of waypoints representing its future trajectory. The core features of DIFFTEST4AV require only the ability to estimate the confidence that an output is an outlier and measure the outlier's severity. Given the specific semantics of, e.g., a trajectory output, specialized techniques [52], [53] would need to be adapted to calculate these two values.

Out-of-Distribution Inputs. Our approach allows for arbitrary input data to be used for differential testing. Some of this data could reside outside the system's expected operational design domain (ODD) [54]. For example, Figure 6 shows an identified failure at 97% confidence and almost 30° severity in which the system is approaching an intersection. However, the specifications for the system indicate that the system is not intended to handle intersections [55]. As such, this identification would be a false positive. However, recent work has presented methods to automate the process of ODD detection, which could be used to reduce the number of false positives [56].

Strength of Pseudo-oracles. Finally, we recognize several limitations of the differential oracle. When all AVs violate the same specification, the differential oracle cannot detect the failure. This limitation can be mitigated through the use of more and varied reference systems. Setting the confidence, severity, and duration thresholds too high may either overlook differences indicative of a failure, whereas a threshold set too low could lead to false positives. Last, if there are many correct answers, the SUT may pick a correct answer, but one that is unique from the reference systems. This will result in a false-positive failure identification, but can similarly addressed through more and varied reference systems.

IV. STUDY

We pose the following research questions to explore DIFFTEST4AV's ability to identify inputs leading to failures:

RQ1: To what extent can DIFFTEST4AV identify individual inputs leading to high-confidence failures?
RQ2: To what extent can DIFFTEST4AV identify individual inputs leading to high-impact failures?
RQ3: To what extent can DIFFTEST4AV identify input sequences leading to high-impact, long-running failures?

TABLE III: Reference Systems and SUT Evaluated

System Label	Date	Commit ID
AV^1	Apr. 2022	5159878 [57]
AV^2	Jul. 2022	b51a90b [58]
AV^3	Nov. 2022	a48ec65 [59]
AV^4	Mar. 2023	cb2a53a [60]
AV^{SUT}	Jun. 2023	2ebd7ab [61]

TABLE IV: Datasets Utilized

Label	# Videos	Duration	# Input Images
comma.ai 2016	11	7 hrs	391,843
comma.ai 2k19	2035	34 hrs	1,825,111
External JUtah	50	43 hrs	2,362,708

A. Setup

Conducting the study required a target SUT, reference systems, and input datasets. We describe our choices next.

Systems. We selected comma.ai's OpenPilot as our target AV. This commercial, open-source, road-deployed system is capable of performing various tasks including Automatic Lane Centering (ALC), Adaptive Cruise Control (ACC), Lane Departure Warning (LDW), and Forward Collision Warning (FCW). While OpenPilot is more accurately characterized as an Automated Driving System (ADS) than an AV, the ALC capability is a crucial component of any AV system. OpenPilot is compatible with over 250 vehicle models [62] and has reportedly driven over 50 million miles in deployment [63], demonstrating the maturity of this safety-critical system. We analyzed five versions of OpenPilot's open-source ALC [57]–[61], listed in Table III; the latest version served as the SUT while the four prior versions were the reference systems. These versions are spaced roughly three months apart, allowing sufficient time for differences to develop between each iteration. All versions have consistent specifications and rely on camera-based inputs to determine the steering angle of the vehicle.

Datasets. We used three datasets of images collected from real-world driving to evaluate DIFFTEST4AV's ability to use arbitrary data to generate tests. The first two datasets are from comma.ai to illustrate the presence of viable inputs within datasets used by the same company that produced the AV that can be transformed into test cases. We leverage the comma.ai 2016 dataset [64], consisting of 11 videos totaling 7 hours, and the comma.ai 2k19 dataset [65], with 2035 videos totaling 34 hours. These selections were based on the premise that data from the same source as the SUT would likely adhere to the system's specifications and exhibit minimal, yet potentially some, failures. To assess our approach's ability to utilize arbitrary data from other sources of real-world sensor data, we examined the most recent 50 videos, totaling 43 hours, from the JUtah dashcam video collection [66], unaffiliated with comma.ai (labeled "External JUtah" in the results to make clear this lack of affiliation). This collection, which includes thousands of dashcam recordings, showcases a wide range of publicly accessible sensor data. Due to its lack of affiliation with comma.ai, we anticipated a higher incidence of failures in this dataset. The three datasets are summarized in Table IV. Since the three datasets lack velocity state information, which OpenPilot uses to compute its output, we pair each image with a set velocity of 30 mph, a reasonable average for the variety of scenarios shown in the datasets. We use the identify function for the behavior function, b, discussed in Section III-D. In this case, b returns the steering angle command of the OpenPilot system as the estimate of the future system behavior; estimating future position would require the AV's velocity, which is not available. Thus, we present the results in units of steering angle to ease interpretability. Exploring alternative approaches to setting the state, how the state affects our approach, and the utility of richer b functions is left for future work.

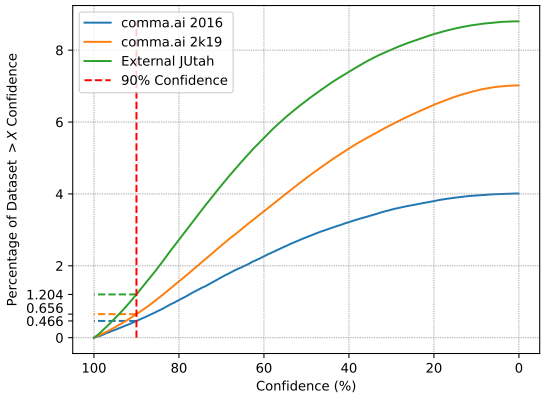
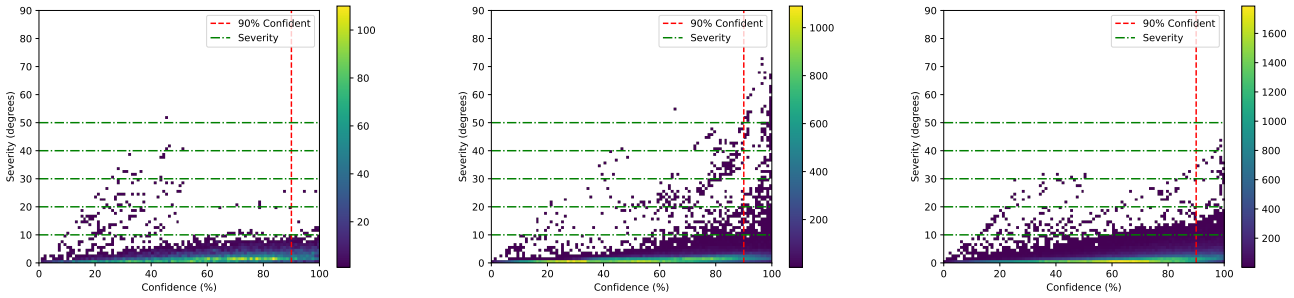


Fig. 7: Percentage of dataset above a given confidence (note reversed X-axis)

B. RQ1: To what extent can DIFFTEST4AV identify individual inputs leading to high-confidence failures?

Given a dataset of inputs, a set of reference systems, and the SUT, the first step of DIFFTEST4AV aims to identify high-confidence failures by assigning each test a score that defines the confidence that the SUT's output represents a failure for that input. In this question we investigate the distribution of confidences identified by DIFFTEST4AV for the datasets. If the SUT was always the best system, the distribution would contain only 0% confidence inputs (ideal for the SUT). If the SUT always disagreed with the reference systems and they all fully agreed with each other, the distribution would contain only 100% confidence inputs (ideal for DIFFTEST4AV's ability to find failures). However, in practice we expect that, for a robust system, most of the inputs will be at 0% confidence, with a relatively small number of inputs from the "long tail" leading to high-confidence failures.

Figure 7 shows the cumulative distribution of the confidence of failure achieved for the corresponding percentage of each dataset. Note that the X-axis is reversed so that the graph is presented as monotonically increasing; as the confidence threshold decreases, more of the dataset is included above that threshold. For each dataset, less than 10% of the dataset has $> 0\%$ confidence. Recall from Section III, if the SUT is not the most extreme output, then its confidence is defined to be 0. This indicates that less than 10% of the data meet the



(a) comma.ai 2016 Failure Histogram (b) comma.ai 2k19 Failure Histogram (c) External JUtah Failure Histogram

Fig. 8: Severity vs Confidence histograms across the three datasets. Best viewed on a screen.

TABLE V: Number of frames (# F) and videos (# V) leading to failures at 90% confidence and varying severity thresholds

	comma.ai 2016						comma.ai 2k19						External JUtah					
	$> conf$		$> sev$		Both		$> conf$		$> sev$		Both		$> conf$		$> sev$		Both	
sev	# F	# V	# F	# V	# F	# V	# F	# V	# F	# V	# F	# V	# F	# V	# F	# V	# F	# V
10°			183	10	16	3			977	45	485	30			1231	35	530	22
20°			67	8	2	1			494	19	263	12			124	13	49	4
30°	1826	11	18	6	0	0	11967	1007	310	11	174	6	28442	45	33	7	23	2
40°			5	3	0	0			183	8	140	6			3	1	3	1
50°			1	1	0	0			82	6	81	5			0	0	0	0

necessary condition for finding a failure. Also, comma.ai 2016 has the lowest percentage of non-zero-confidence inputs at just under 4%, followed by comma.ai 2k19, and finally External JUtah. This is expected as comma.ai 2016 is the oldest internal dataset, so it may have already been used to refine OpenPilot; similarly for the 2k19 dataset, except it is both larger and newer, perhaps explaining its higher percentage. Finally, we see that External JUtah has the largest percentage of the dataset at all levels of confidence as we expected as it is external to comma.ai and not used for development.

These trends also hold at high confidence values. At 90% confidence, marked with the dashed line in Figure 7, External JUtah has almost double the proportion of the next closest in comma.ai 2k19, with 1.20% leading to $> 90\%$ confidence failures compared to 0.66%. At the 90% confidence threshold, DIFFTEST4AV can identify the 42,235 high-confidence failures among 4,579,662 inputs across these datasets (0.92%), and at a threshold of 99% it identifies the 3,284 (0.07%) high-confidence failures.

RQ1 Findings: DIFFTEST4AV effectively identifies individual sensor inputs leading to high-confidence failures across large-scale datasets. From 4.58 million total inputs, it uncovered 42,235 high-confidence failures at 90% confidence, and 3,284 at 99% confidence. This emphasizes DIFFTEST4AV's ability to extract rare but critical failures from vast datasets. Additionally, external datasets yielded the highest proportion of failures, demonstrating DIFFTEST4AV 's utility in leveraging diverse and previously untapped data.

C. *RQ2: To what extent can* DIFFTEST4AV *identify individual inputs leading to high-impact failures?*

Now that DIFFTEST4AV has found high-confidence failures in the SUT, we can further leverage it to identify inputs that lead to high-confidence, high-severity, i.e. high-impact, failures. In this RQ we focus on the ability to identify single-instant failures; the next RQ will analyze multi-frame failures.

Figure 8 provides a 2D histogram of the number of inputs that lead to a failure at various thresholds of severity and confidence across the three datasets (note that 0% confidence inputs are not shown since those are dropped by the confidence module). To further examine the distribution of high-confidence and high-severity failures, we set a red vertical line marking 90% confidence and green horizontal lines marking 10° to 50° severities. These are further explored in Table V, which describes, for the different thresholds, the number of high-confidence inputs (right of the red line), the number of high-severity inputs (above the green line), and the number of inputs that are both (upper right), and thus high-impact. The table shows the number of single-frame inputs (# F) and the number of videos (# V) from which those frames originated.

While RQ1 demonstrated that the confidence values spanned the full-range, here we observe that the overwhelming majority of non-zero confidence inputs lead to low-severity failures. Figure 8a shows that most high-severity failures for the comma.ai 2016 dataset reside in the 10%-50% confidence range. Table V further highlights that there are no high-impact failures above 30°, and only 2 above 20°. These 2 inputs represent 0.005% of the dataset. We hypothesize that this low failure rate is due to comma.ai 2016 being an older, internal dataset that the developers of the SUT use.

Figure 8b and Table V show that the comma.ai 2k19 provides many more chances to identify high-confidence and high-severity failures than the other datasets. This is unexpected, and we hypothesize that this may arise from the diversity of the dataset. As shown in Table IV, the comma.ai 2k19 dataset contains over 2000 separate videos; this may indicate that the dataset better covers the long-tail distribution of possible inputs and thus is better able to exhibit failures. As shown in Table V, the comma.ai 2k19 dataset is the only dataset to observe high-impact failures above 90% confidence and 50° severity, with 81 failure-inducing frames (0.05%) identified across 5 videos (0.25%). If high-impact failures were found at a rate of 0.25% of videos for the other datasets, then we would expect to find one failure per 400 videos. With comma.ai 2016 and External JUtah having 11 and 50 videos respectively, their diversity may not be sufficient to uncover these high-impact failures.

Although External JUtah does not find high-impact failures at the 50° severity threshold, Figure 8c and Table V show that External JUtah does contain inputs that lead to failures at 40°, and also has a comparable number of failures to comma.ai 2k19 at the 10° severity threshold. Although External JUtah is slightly larger than comma.ai 2k19, it has fewer but longer videos. These results suggest that in the future, testing datasets sourced from arbitrary data should focus on variety to increase the chances of finding failures of the highest severity.

DIFFTEST4AV can identify the 143 (0.003%) inputs that yield high-impact failures at over 90% confidence and 40° severity, and the 54 (0.001%) inputs that yield high-impact failures at over 99% confidence and 50° severity, highlighting its ability to automatically identify these rare, but critical high-impact failures.

RQ2 Findings: From the high-confidence failure-inducing inputs identified in RQ1, DIFFTEST4AV further filters to isolate the most critical high-severity, high-impact failures. At a 50° severity threshold, it identified 81 high-impact failures at 90% confidence and 54 high-impact failures at 99% confidence. These rare but significant failures highlight DIFFTEST4AV's capability to pinpoint critical issues with both high severity and high confidence across diverse datasets.

D. RQ3: To what extent can DIFFTEST4AV *identify input sequences leading to high-impact, long-running failures?*

While RQ2 identified high-confidence and high-severity failures based on a single input, in the AV domain we are concerned with sequences of inputs that all indicate a failure as those are more likely to lead to system level failures.

Figure 9 illustrates an example multi-frame failure from the External JUtah dataset. The full 5 frames shown have a minimum severity of 19.8° and a combined confidence of 77.7%. This scene depicts the AV driving toward a curve to the left. Each of the reference systems produce an AV action of straight or slightly left, with the systems indicating harder left turns as the sequence progresses. However, throughout the sequence, the SUT consistently produces an action of turning significantly to the right. If the system were to actuate this behavior, it could pull the AV off the road and prevent it from making the turn, leading to a system-level failure. This highlights the safety-critical nature of these systems and demonstrates DIFFTEST4AV's ability to find long-running, high-impact failures that could lead to system-level failures.

For each dataset, with the SUT operating at $15Hz$, Table VI shows the maximal duration failure over given severity and confidence thresholds using the more conservative estimation approach described in Section III-E. For example, the comma.ai 2k19 dataset, at 50° severity and 90% confidence, had a maximal duration of 27 inputs ($27/15Hz = 1.8$ seconds). This 90% confidence means that the product of the confidences of all 27 separate failures is over 90% and the 50° severity means that all 27 inputs led to failures of at least 50°. At 35mph a vehicle would travel over 90 feet in 1.8 seconds, and a continuous severity over 50° for 27 frames would drastically alter the vehicle's trajectory in that time.

Table VI overall trends are as expected; as the confidence and severity each increase (more stringent failure requirements), the maximal duration failure sequence identified by DIFFTEST4AV decreases. More interesting, Table VI shows that DIFFTEST4AV was able to identify, for each of the datasets, a continuous failure of more than 10 frames at 50% confidence and 10° severity with a peak of a 72-input continuous failure sequence in comma.ai 2k19 dataset for a duration of 4.8 seconds. At 99% confidence and 10° severity, DIFFTEST4AV was still able to identify failures for all datasets that lasted for at least 4 frames. We continue to observe the same trend as with RQ2, with comma.ai 2k19 and External JUtah having substantially more failure-inducing sequences than comma.ai 2016. Further, continuing the trend from RQ2 that we hypothesize is related to dataset diversity, comma.ai 2k19 has longer high-confidence and high-severity failures at all levels.

RQ3 Findings: DIFFTEST4AV effectively identifies long-running failure sequences, critical in the AV domain for their potential to cause system-level failures. At 90% confidence and 50° severity, it detected a maximum sequence of 27 frames (1.8 seconds) of persistent failure behavior. Even at stricter confidence thresholds (99% confidence, 10° severity), DIFFTEST4AV identified failure sequences lasting at least 4 frames across all datasets. These findings demonstrate its capability to detect rare, critical, and extended-duration failures.

E. Threats to Validity

The external validity of our study findings is affected by our choice of AVs to serve as reference systems and SUT. We chose the comma.ai OpenPilot system for study as it is commercially available and actively in-use on public roads. Further, as an open-source system, multiple prior versions

(a) Frame 3465 $C = 99.3\%, S = 22.2°$ (b) Frame 3466 $C = 98.4\%, S = 19.8°$ (c) Frame 3467 $C = 96.9\%, S = 20.8°$ (d) Frame 3468 $C = 93.4\%, S = 20.8°$ (e) Frame 3469 $C = 87.5\%, S = 18.1°$

Fig. 9: Mutli-frame failure found. The sequence has a combined confidence of 77.7% and minimum severity of 19.8°.

TABLE VI: Duration (in frames) of maximal failure per dataset at varying confidence and severity thresholds

	comma.ai 2016 *conf*					comma.ai 2k19 *conf*					External JUtah *conf*				
sev	50%	75%	90%	95%	99%	50%	75%	90%	95%	99%	50%	75%	90%	95%	99%
10°	12	10	9	7	4	72	58	34	22	14	27	17	12	9	4
20°	4	3	2	1	0	64	56	34	22	14	23	17	10	7	3
30°	1	0	0	0	0	52	48	34	20	14	15	11	8	6	3
40°	1	0	0	0	0	42	42	27	19	11	3	3	3	2	0
50°	0	0	0	0	0	33	33	27	19	5	0	0	0	0	0

were publicly available for study to serve as reference systems. Other AVs may present different failure modes, and the quantity and type of failures identified will vary based on the reference systems utilized. However, we believe that the setup of this study as a regression test, in which previous versions of a system are utilized as reference versions for the most recent version as the SUT, represents a prototypical use case for the framework. Our study's external validity is further affected by the datasets choices. We chose three datasets to represent two typical use cases: two internal datasets used for developing and testing the system, and an external dataset to demonstrate our framework's ability to use arbitrary data for testing. However, the long-tail distribution of data means that these datasets may not be representative; future work should conduct further study on additional data.

The internal validity of our study is affected by the complex experimental setup involved in executing the reference systems and SUTs on the test input to evaluate their output. While we extended the open-source code for evaluating the systems provided by comma.ai, there are several complex components including input normalization to resize the input images and match input frame rates. To mitigate this threat, we release an open-source artifact to execute the SUTs and analyze their results[3]. The internal validity of our study is further affected by our execution of the SUT in an open-loop manner. Although offline open-loop testing is commonly utilized [67] due to its safety benefits, it fundamentally limits our ability to reason about whether continuous failures in an open-loop context would translate to real-world failures in a closed-loop context. To mitigate this threat, DIFFTEST4AV is conservative in its estimation and can be further tuned to identify only the most high-confidence, high-severity failures.

[3]https://github.com/less-lab-uva/DiffTest4AV
Archived at: https://archive.softwareheritage.org/browse/origin/https://github.com/less-lab-uva/DiffTest4AV

V. CONCLUSION

In this work we address the unique challenges of testing AVs while tapping into increasingly vast sensor datasets. We propose a differential testing approach, DIFFTEST4AV, which is able to leverage reference systems to identify high-confidence, high-severity, long-running failures in an SUT that can lead to system-level failures—a critical need given the increasing complexity of AV systems and their deployment contexts. Our evaluation using multiple versions of comma.ai's OpenPilot showed DIFFTEST4AV's capability to detect significant failures utilizing inputs from internal and external arbitrary sensor datasets. Overall, DIFFTEST4AV processed over 4,579,662 inputs to identify 81 (0.002%) high-impact failures judged at 90% confidence and 50° severity, showcasing its ability to find the high-impact AV failure needle in the arbitrary data haystack. Notably in terms of duration, in evaluating the comma.ai 2k19 dataset, DIFFTEST4AV identified with 90% confidence, 27 consecutive input images that each caused failures of over 50° severity. This would result in significant changes in behavior and could result in potentially fatal failures for an AV. These findings indicate that DIFFTEST4AV successfully identified high-confidence and high-severity failures, highlighting its potential to enhance AV testing and safety. Moving forward, we will extend this work by incorporating additional AV systems and datasets, further refining our confidence and severity metrics, and exploring the application of DIFFTEST4AV in closed-loop testing environments to better simulate real-world conditions.

ACKNOWLEDGMENTS

This work was supported in part by NSF Awards #2403060 and #2312487. Trey Woodlief was supported by a University of Virginia SEAS Fellowship. We would like to extend our sincere thanks to comma.ai for making OpenPilot available, which made this research possible.

REFERENCES

[1] P. McCausland, "Self-driving uber car that hit and killed woman did not recognize that pedestrians jaywalk," 2019, [Online; accessed 09-July-2023]. [Online]. Available: https://www.nbcnews.com/tech/tech-news/self-driving-uber-car-hit-killed-woman-did-not-recognize-n1079281

[2] Jackie Wattles, "CNN Business - Tesla on Autopilot crashed when the driver's hands were not detected on the wheel," https://www.cnn.com/2019/05/16/cars/tesla-autopilot-crash/index.html, 2019, [Online; accessed 09-July-2023].

[3] T. Krisher, "Us report: Nearly 400 crashes of automated tech vehicles," 2022, [Online; accessed 22-July-2023]. [Online]. Available: https://apnews.com/article/self-driving-car-crash-data-ae87cadec79966a9ba56e99b4110b8d6

[4] B. Pietsch, "2 killed in driverless tesla car crash, officials say," 2021, [Online; accessed 09-July-2023]. [Online]. Available: https://www.nytimes.com/2021/04/18/business/tesla-fatal-crash-texas.html

[5] IEEE Connected Vehicles, "Google reports self-driving car disengagements," https://site.ieee.org/connected-vehicles/2015/12/15/google-reports-self-driving-car-disengagements/, 2015, [Online; accessed 09-July-2023].

[6] O. of Public Affairs, "Av permit holders report close to 2 million test miles in california," https://www.dmv.ca.gov/portal/news-and-media/av-permit-holders-report-close-to-2-million-test-miles-in-california/, 2021.

[7] ——, "Av permit holders report 4 million test miles in california," https://www.dmv.ca.gov/portal/news-and-media/av-permit-holders-report-4-million-test-miles-in-california/, 2022.

[8] ——, "Autonomous vehicle permit holders report 5.7 million test miles in california," https://www.dmv.ca.gov/portal/news-and-media/autonomous-vehicle-permit-holders-report-5-7-million-test-miles-in-california/, 2023.

[9] ——, "Autonomous vehicle permit holders report a record 9 million test miles in california in 12 months," https://www.dmv.ca.gov/portal/news-and-media/news-releases/autonomous-vehicle-permit-holders-report-a-record-9-million-test-miles-in-california-in-12-months/, 2024.

[10] G. Jahangirova, A. Stocco, and P. Tonella, "Quality metrics and oracles for autonomous vehicles testing," in 2021 14th IEEE conference on software testing, verification and validation (ICST). IEEE, 2021, pp. 194–204.

[11] comma.ai, "Github - openpilot," https://github.com/commaai/openpilot, 2024.

[12] Teslascope, "Full self-driving," https://teslascope.com/software/full-self-driving, 2024.

[13] L. Baresi and M. Young, "Test oracles," 2001.

[14] M. Zalewski, American Fuzzy Lop, 2013 (accessed October 27, 2020), https://lcamtuf.coredump.cx/afl/.

[15] P. Godefroid, M. Y. Levin, and D. Molnar, "Sage: whitebox fuzzing for security testing," Queue, vol. 10, no. 1, pp. 20–27, 2012.

[16] G. Grieco, M. Ceresa, and P. Buiras, "Quickfuzz: An automatic random fuzzer for common file formats," ACM SIGPLAN Notices, vol. 51, no. 12, pp. 13–20, 2016.

[17] D. J. Richardson, S. L. Aha, and T. O. O'malley, "Specification-based test oracles for reactive systems," in Proceedings of the 14th international conference on Software engineering, 1992, pp. 105–118.

[18] P. Stocks and D. Carrington, "A framework for specification-based testing," IEEE Transactions on software Engineering, vol. 22, no. 11, pp. 777–793, 1996.

[19] R. M. Hierons, K. Bogdanov, J. P. Bowen, R. Cleaveland, J. Derrick, J. Dick, M. Gheorghe, M. Harman, K. Kapoor, P. Krause et al., "Using formal specifications to support testing," ACM Computing Surveys (CSUR), vol. 41, no. 2, pp. 1–76, 2009.

[20] V. S. Alagar, K. Periyasamy, and K. Periyasamy, Specification of software systems. Springer, 2011.

[21] W. M. McKeeman, "Differential testing for software," Digital Technical Journal, vol. 10, no. 1, pp. 100–107, 1998.

[22] D. Coelho and M. Oliveira, "A review of end-to-end autonomous driving in urban environments," IEEE Access, vol. 10, pp. 75 296–75 311, 2022.

[23] Z. Zhong, G. Kaiser, and B. Ray, "Neural network guided evolutionary fuzzing for finding traffic violations of autonomous vehicles," IEEE Transactions on Software Engineering, 2022.

[24] L. Klampfl, F. Klück, and F. Wotawa, "Using genetic algorithms for automating automated lane-keeping system testing," Journal of Software: Evolution and Process, vol. 36, no. 3, p. e2520, 2024.

[25] C. Stark, C. Medrano-Berumen, and M. İ. Akbaş, "Generation of autonomous vehicle validation scenarios using crash data," in 2020 SoutheastCon. IEEE, 2020, pp. 1–6.

[26] D. Gandhi, L. Pinto, and A. Gupta, "Learning to fly by crashing," in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017, pp. 3948–3955.

[27] 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.

[28] M. von Stein, D. Shriver, and S. Elbaum, "Deepmaneuver: Adversarial test generation for trajectory manipulation of autonomous vehicles," IEEE Transactions on Software Engineering, 2023.

[29] S. Bak, J. Betz, A. Chawla, H. Zheng, and R. Mangharam, "Stress testing autonomous racing overtake maneuvers with rrt," in 2022 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2022, pp. 806–812.

[30] L. Feng, Q. Li, Z. Peng, S. Tan, and B. Zhou, "Trafficgen: Learning to generate diverse and realistic traffic scenarios," in 2023 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2023, pp. 3567–3575.

[31] S. Konur, C. Dixon, and M. Fisher, "Formal verification of probabilistic swarm behaviours," in Swarm Intelligence: 7th International Conference, ANTS 2010, Brussels, Belgium, September 8-10, 2010. Proceedings 7. Springer, 2010, pp. 440–447.

[32] H. Liang, J. S. Dong, J. Sun, and W. E. Wong, "Software monitoring through formal specification animation," Innovations in Systems and Software Engineering, vol. 5, pp. 231–241, 2009.

[33] H. Bhuiyan, G. Governatori, A. Bond, and A. Rakotonirainy, "Traffic rules compliance checking of automated vehicle maneuvers," Artificial Intelligence and Law, vol. 32, no. 1, pp. 1–56, 2024.

[34] Y. Sun, C. M. Poskitt, X. Zhang, and J. Sun, "Redriver: Runtime enforcement for autonomous vehicles," in Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, 2024, pp. 1–12.

[35] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T. Tse, and Z. Q. Zhou, "Metamorphic testing: A review of challenges and opportunities," ACM Computing Surveys (CSUR), vol. 51, no. 1, pp. 1–27, 2018.

[36] S. Segura, G. Fraser, A. B. Sanchez, and A. Ruiz-Cortés, "A survey on metamorphic testing," IEEE Transactions on software engineering, vol. 42, no. 9, pp. 805–824, 2016.

[37] Z. Q. Zhou and L. Sun, "Metamorphic testing of driverless cars," Communications of the ACM, vol. 62, no. 3, pp. 61–67, 2019.

[38] 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 Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, 2018, pp. 132–142.

[39] X. Gao, R. K. Saha, M. R. Prasad, and A. Roychoudhury, "Fuzz testing based data augmentation to improve robustness of deep neural networks," in 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE). IEEE, 2020, pp. 1147–1158.

[40] 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.

[41] C. Sakaridis, D. Dai, and L. Van Gool, "Semantic foggy scene understanding with synthetic data," International Journal of Computer Vision, vol. 126, no. 9, pp. 973–992, 2018.

[42] T. Woodlief, S. Elbaum, and K. Sullivan, "Semantic image fuzzing of ai perception systems," in Proceedings of the 44th International Conference on Software Engineering, 2022, pp. 1958–1969.

[43] G. Christian, T. Woodlief, and S. Elbaum, "Generating realistic and diverse tests for lidar-based perception systems," in 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE). IEEE, 2023, pp. 2604–2616.

[44] A. Groce, G. Holzmann, and R. Joshi, "Randomized differential testing as a prelude to formal verification," in 29th International Conference on Software Engineering (ICSE'07). IEEE, 2007, pp. 621–631.

[45] D. M. Schwarz, L. Rolland, and J. B. Johnston, "Identifying real-world problems with automated vehicles by detecting behavioral differences in steering movements between the human driver and machine," in 2022 IEEE 28th International Conference on Engineering, Technology and Innovation (ICE/ITMC) & 31st International Association For Management of Technology (IAMOT) Joint Conference. IEEE, 2022, pp. 1–9.

[46] H. Wang, M. J. Bah, and M. Hammad, "Progress in outlier detection techniques: A survey," Ieee Access, vol. 7, pp. 107 964–108 000, 2019.

[47] W. J. Dixon, "Ratios involving extreme values," The Annals of Mathematical Statistics, vol. 22, no. 1, pp. 68–78, 1951.

[48] F. E. Grubbs, "Procedures for detecting outlying observations in samples," Technometrics, vol. 11, no. 1, pp. 1–21, 1969.

[49] E. Barini, "Features and performance of some outlier detection methods," Journal of Applied Statistics, vol. 38, no. 10, pp. 2133–2149, 2011.

[50] M. Chernick, "Note on the robustness of dixon's ratio test in small samples.[testing for outliers]," Oak Ridge National Lab., TN (USA), Tech. Rep., 1980.

[51] comma.ai Team, "Development speed over everything," https://blog.comma.ai/dev-speed/#continuous-integration-testing, 2022.

[52] T. Laurent, S. Klikovits, P. Arcaini, F. Ishikawa, and A. Ventresque, "Parameter coverage for testing of autonomous driving systems under uncertainty," ACM Transactions on Software Engineering and Methodology, vol. 32, no. 3, pp. 1–31, 2023.

[53] M. L. Dias, C. L. C. Mattos, T. L. da Silva, J. A. F. de Macedo, and W. C. Silva, "Anomaly detection in trajectory data with normalizing flows," in 2020 international joint conference on neural networks (IJCNN). IEEE, 2020, pp. 1–8.

[54] "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," April 2021. [Online]. Available: http://dx.doi.org/10.4271/J3016_202104

[55] comma.ai, "openpilot," https://github.com/commaai/openpilot/blob/b816b5b/docs/LIMITATIONS.md, 2023.

[56] C. Hildebrandt, T. Woodlief, and S. Elbaum, "ODD-diLLMma: Driving Automation System ODD Compliance Checking using LLMs," in 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2024.

[57] comma.ai, "openpilot 5159878," 2022. [Online]. Available: https://github.com/commaai/openpilot/commit/515987838908c1a4f5c822919ccf2d78ebac144b

[58] ——, "openpilot b51a90b," 2022. [Online]. Available: https://github.com/commaai/openpilot/commit/b51a90b5a87e0b6388191f7cc5857af8d72e79de

[59] ——, "openpilot a48ec65," 2022. [Online]. Available: https://github.com/commaai/openpilot/commit/a48ec655ac4983145bc93c712ecabac75b886e11

[60] ——, "openpilot cb2a53a," 2023. [Online]. Available: https://github.com/commaai/openpilot/commit/cb2a53ae80ab3917986266290f37ef0228a6ca21

[61] ——, "openpilot 2ebd7ab," 2023. [Online]. Available: https://github.com/commaai/openpilot/commit/2ebd7ab088ade61bbf661c140483fc477d444bc2

[62] ——, "Openpilot supports 250+ vehicles," https://comma.ai/vehicles, 2023.

[63] ——, "Media," https://www.comma.ai/media, 2023.

[64] E. Santana and G. Hotz, "Learning a driving simulator," arXiv preprint arXiv:1608.01230, 2016.

[65] H. Schafer, E. Santana, A. Haden, and R. Biasini, "A commute in data: The comma2k19 dataset," arXiv preprint arXiv:1812.05752, 2018.

[66] JUtah, "Driving around the world, 30+ countries," https://www.youtube.com/@jutah, December 2023.

[67] H. Araujo, M. R. Mousavi, and M. Varshosaz, "Testing, validation, and verification of robotic and autonomous systems: a systematic review," ACM Transactions on Software Engineering and Methodology, vol. 32, no. 2, pp. 1–61, 2023.