Testing of Autonomous Driving Systems: Where Are We and Where Should We Go?

A study of common practices, needs and a way forward

Guannan Lou
Macquarie University
Sydney, NSW, Australia
guannan.lou@mq.edu.au

Yao Deng
Macquarie University
Sydney, NSW, Australia
yao.deng@hdr.mq.edu.au

Xi Zheng*
Macquarie University
Sydney, NSW, Australia
james.zheng@mq.edu.au

Mengshi Zhang*
Meta
Menlo Park, CA, USA
mengshizhang@fb.com

Tianyi Zhang
Purdue University
West Lafayette, IN, USA
tianyi@purdue.edu

ABSTRACT

Autonomous driving has shown great potential to reform modern transportation. Yet its reliability and safety have drawn a lot of attention and concerns. Compared with traditional software systems, autonomous driving systems (ADSs) often use deep neural networks in tandem with logic-based modules. This new paradigm poses unique challenges for software testing. Despite the recent development of new ADS testing techniques, it is not clear to what extent those techniques have addressed the needs of ADS practitioners. To fill this gap, we present the first comprehensive study to identify the current practices and needs of ADS testing. We conducted semi-structured interviews with developers from 10 autonomous driving companies and surveyed 100 developers who have worked on autonomous driving systems. A systematic analysis of the interview and survey data revealed 7 common practices and 4 emerging needs of autonomous driving testing. Through a comprehensive literature review, we developed a taxonomy of existing ADS testing techniques and analyzed the gap between ADS research and practitioners' needs. Finally, we proposed several future directions for SE researchers, such as developing test reduction techniques to accelerate simulation-based ADS testing.

KEYWORDS

Autonomous Driving, Software Testing, Empirical Study

ACM Reference Format:
Guannan Lou, Yao Deng, Xi Zheng, Mengshi Zhang, and Tianyi Zhang. 2022. Testing of Autonomous Driving Systems: Where Are We and Where Should We Go?: A study of common practices, needs and a way forward. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '22), November 14–18, 2022, Singapore, Singapore.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3540250.3549111

*Corresponding authors: Xi Zheng, Mengshi Zhang.

ESEC/FSE '22, November 14–18, 2022, Singapore, Singapore

ACM ISBN 978-1-4503-9413-0/22/11…$15.00
https://doi.org/10.1145/3540250.3549111

1 INTRODUCTION

Autonomous driving has been making great strides towards reality in recent years. In 2017, Waymo launched the trail of fully autonomous ride-hailing services in California [1]. More recently, Tesla released the beta version of its Full Self-Driving (FSD) software, which has been installed on at least 60K Tesla vehicles [2]. However, given the traffic accidents caused by autonomous vehicles [3–5] , there is still a long way to ensure the robustness and reliability of autonomous driving systems (ADSs). Similar to traditional software systems, autonomous driving companies also adopt testing as the main quality assurance mechanism for ADSs. Several leading companies such as Waymo and Tesla have their own fleets to perform extensive on-road testing. Furthermore, simulation testing environments such as Carla [6] are widely adopted to test various driving scenarios or extreme conditions that cannot be easily replicated in reality.

In recent years, the Software Engineering (SE) community has also proposed many testing techniques to improve the safety and reliability of ADSs [7–48]. For example, DeepRoad and DeepTest [22, 23] leverage metamorphic testing to test ADSs under extreme weather conditions. AC3R [27] generates critical driving scenarios (e.g. collisions) in a simulation environment based on crash reports. These methods have shown promising results for end-to-end driving models. However, with the rapid evolution of autonomous driving technologies, industrial ADSs have become much more sophisticated these days, using multiple perception models in tandem with logic-based control and planning modules. Currently, there is a lack of comprehensive understanding of the emerging needs of ADS testing and to what extent existing ADS testing techniques meet the needs of ADS practitioners. To bridge this gap, we adopted a mixed methods research design [49] with a combination of qualitative interview study, large-scale survey, and literature review to investigate the following three research questions:

- **RQ1**. *What are the industrial practices of ADS testing?*
- **RQ2.** *What are the emerging needs of ADS testing?*

- **RQ3.** *To what extent existing ADS testing techniques address the industrial needs?*

We first interviewed developers from 10 autonomous driving companies to collect qualitative responses on ADS testing practices and needs. Based on the insights from the interview study, we developed a survey and solicited quantitative responses from a boarder audience. Specifically, we sent out 1978 surveys to ADS developers and received 100 complete and valid responses. Through comprehensive data analysis and triangulation, we summarized seven common practices in ADS development and testing, some of which have not been accounted by existing testing techniques. For example, ADS practitioners adopt more system-level testing metrics such as consistency and latency when testing an ADS, rather than just using model accuracy. Furthermore, we identified four common needs of ADS testing: (1) *identifying possible corner cases and unexpected driving scenarios*, (2) *speeding up ADS testing*, (3) *tool support for constructing complex driving scenarios*, and (4) *tool support for data labeling*.

We further conducted literature review on ADS testing methods proposed by the SE community. We manually went through 117 papers that mentions autonomous driving or related keywords in 28 SE conferences and identified 42 papers specifically about ADS testing. We categorized them and developed a taxonomy of different kinds of research on ADS testing. We identified four gaps between existing research and industrial needs. For instance, existing ADS testing methods only consider simple image transformations. There is a lack of support for identifying richer and more complex traffic scenarios that ADS developers really need in practice. Based on these gaps, we proposed several future directions. For example, SE researchers can leverage test selection and prioritization techniques to speed up ADS testing (Need 2). Specifically, to account for the multi-module architecture of ADSs, test selection methods can be formulated as a multi-objective optimization problem to maximize multiple coverage metrics for different modules (e.g., neural coverage for a perception model and statement coverage for a logic-based control module), rather than a single model.

Paper organization. Section 2 discusses related work in ADS testing. Section 3 illustrates how we conduct interviews and surveys. Section 4 provides current practice of ADS testing in industry. Section 5 summarizes needs of industry in ADS testing. Section 6 discusses the existing SE solutions to these needs and future research directions. Section 7 discusses the threats to validity in this study. Section 8 concludes this work.

2 RELATED WORK

There are several literature reviews on testing and verification of machine learning (ML) models [50–53]. The most comprehensive literature review is by Zhang et al. [50]. They analyzed 114 papers related to machine learning testing and provided an overview of various testing properties, components, and workflows of ML models. In addition, Zhang et al. identified several challenges of testing ML models, such as how to generate natural test inputs. Compared with these studies, our study focuses on ADS testing. In addition to a literature review, we also interviewed and surveyed ADS practitioners to understand the common practices and needs of ADS testing. We found that the unique characteristics of ADSs (e.g., the multi-module architecture) along with the special testing needs pose new challenges and opportunities compared with ordinary ML models, e.g., leveraging multi-objective search in test selection for multi-module ADSs.

Two recent studies analyzed the codebase and bugs in autonomous driving systems [35, 54]. Peng et al. [54] conducted a case study of Baidu Apollo [55] and summarized the architecture and individual modules in the Apollo system. They found that Apollo lacked adequate testing at the system level. Garcia et al. [35] analyzed 16,851 commits and 499 issues in Apollo and Autoware [56]. They classified these issues and summarized their symptoms and root causes. There are also several empirical studies on bugs in ML applications [57–61]. For example, Zhang et al. [58] analyzed 175 software bugs in ML applications built by TensorFlow.

The most related work to our study includes several literature reviews on autonomous driving testing [43, 62, 63]. Huang et al. [62] reviewed testing and verification methods from the Intelligent Vehicle (IV) community. They summarized not only testing methods for individual modules in the software stack, but also hardware-in-the-loop testing methods for hardware components and integrated testing for the entire vehicle. Compared with Huang et al., we focus on testing methods from the software perspective. Our literature review summarized recent advances in ADS testing from the SE community. Please refer to Section 6 for a detailed summary and discussion of these software testing techniques. Koopman and Wagner proposed five challenges of testing autonomous vehicles based on the V model for autonomous vehicles [63]. Masuda described the software architecture of autonomous vehicle simulations and discussed several software testing challenges of such simulations [43]. Compared with them, our discussion is anchored upon common practices and needs of ADS testing from interviews and online surveys with ADS practitioners.

3 METHODOLOGY

Following [64], Our empirical study contains 3 steps, as shown in Figure 1. First, we conducted semi-structured interviews with 10 developers from 10 different autonomous driving companies. Based on the findings from these interviews, we further conducted a large-scale survey with 100 ADS practitioners to quantitatively validate our findings from the interviews. We then summarized and triangulated the findings from the surveys and the interviews. Third, we conducted an literature review of SE papers related to ADS testing. We categorized those papers and developed a taxonomy of ADS testing techniques. By comparing the taxonomy and the emerging needs of ADS practitioners, we identified the research gaps.

3.1 Interviews

Interview protocol. We designed an interview guide[1] for the semi-structured interviews. The interview began with a short introduction of our study. Then, we asked high-level questions about: 1) the background and expertise of interviewees, such as the current job role and how long they have been working on ADSs, 2) the current practices, methodologies, and tools they used for ADS testing, and 3) the challenges and difficulties they faced in ADS testing. We first

[1]The complete interview guide is publicly available here: https://bit.ly/3DKia0C

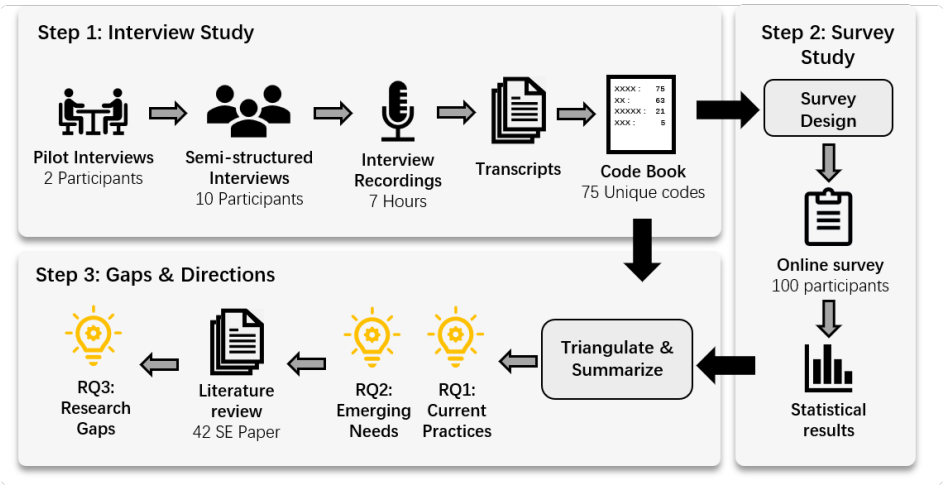

Figure 1: Research methodology

Table 1: Interview participant background

	Experience	Responsibilities
P1	2 years	Recognition algorithm design and testing
P2	2 years	Simulation testing and real-world testing
P3	6 years	Passenger car development
P4	1.5 years	Perception system development
P5	10 years	ADS testing
P6	3 years	ADS testing
P7	1.5 years	Perception system testing
P8	5 years	ADS developing and testing
P9	1.5 years	Recognition algorithm design and testing
P10	1.5 years	ADS testing, Simulation testing

conducted 2 pilot interviews, based on which we further refined the interview guide. Then, we interviewed 10 developers from 10 different autonomous driving companies. Each interview took between 30 minutes and an hour. The interviews were recorded with permission of participants and then transcribed for analysis.

Participants. We recruited 10 software developers from 10 different autonomous driving companies based on our personal network, industrial collaboration, and social media. As shown in Table 1, these participants had at least one and a half years of experience (3.4 years on average) in ADS development and testing. In addition, their responsibilities also covered every aspect of ADS testing, including modular testing, simulation-based testing, and real-world testing.

Analysis. We transcribed a total of 7 hours of interview recordings to text using an audio transcription tool called iflyrec[2]. We manually corrected errors in the transcripts. The first two authors conducted inductive thematic analysis [65][3] using a qualitative data analysis software called MAXQDA[4]. Specifically, they first independently labeled the transcripts to extract relevant or insightful responses from participants and summarize them into short descriptive texts, which are called codes. Then they met each other, compared their codes, and discussed any inconsistencies. They continuously refined the codes during the discussion. Then they grouped related codes into themes. Results of the thematic analysis were regularly reported and discussed with the whole research team. The final inter-rater agreement ratio of the first two authors is 0.85, measured by Cohen's Kappa [66].

3.2 Survey

Since the findings in the interview study are only based on responses from 10 participants, we further conducted a large-scale survey to validate the interview findings and solicit more feedback from a broader population of ADS practitioners.

Survey design. We designed a survey[5] with 33 questions in 5 sections, including *background*, *autonomous driving system*, *testing practices and challenges in ADS*, *testing needs in ADS*, and *follow up*. The *background* section asked about participants' background, expertise, and their current roles in the company or organization. The *autonomous driving system* section asked about the ADSs participants have worked on. The *testing practices and challenges on ADS* section asked about three types of ADS testing identified from the previous interviews, including unit testing, simulation testing, and on-road testing. For each type of testing, we designed multiple-choice questions based on the findings from the interview study. The options in a multiple-choice question were derived from responses of the interview study. Participants can also select "Others" to supplement alternative answers, or select "I don't know" to indicate they have no insights in the particular question. For each type of testing, we also included an open-ended question in the end to solicit additional feedback on what participants would like to have or improve on. The *testing needs in ADS* section listed four common needs identified in the interview study. We designed a linear scale question for each need and asked participants to provide a numeric response to indicate the importance of each need in a 7-point likert scale, from "unimportant at all" (1) to "very important" (7). The

[2]https://www.iflyrec.com/
[3]The maxqda codebook is publicly available here: https://bit.ly/38K9Csk
[4]https://www.maxqda.com/
[5]The complete survey form is publicly available here: https://bit.ly/2WNVV93

follow up section asked participants their contact information and whether they would like to participate in any follow-up study.

Participants. We recruited survey participants in three ways. First, we sent out surveys to the autonomous driving companies where our interview participants came from. Second, we used the APIs provided by Twitter and LinkedIn to scrape the contact information (if any) of developers with profile associated with autonomous driving. Finally, we searched for popular autonomous driving software repositories on Github such as Apollo [55] and DeepDrive [67]. Then, we manually identified the public email address (if any) of those developers contributed to those repositories.

In total, we sent out 1978 surveys and received 114 responses, with a response rate of 5%. We discarded 14 survey responses since they are not complete. In the end, we collected 100 survey responses that are complete for data analysis.[6] 90% of survey participants were male, 8% were female, and 2% did not disclose their gender identity. 54% of them worked in research institutes, 36% worked in technology companies, 7% of participants worked in traditional vehicle manufactures, and 3% of participants were self-employed. Regarding their job roles, 33% of participants were in R&D positions, 14% were in management positions, and 24% were engineers, including safety engineers, software engineers and perception engineers. 40% of participants had two to three years of working experience in ADS, 29% had worked in ADS for more than three years, and 31% had less than two years of experience.

Analysis. For each multiple-choice question, we plotted the choices made by survey participants in a histogram, such as Figure 2. Specifically, if a survey participant supplemented an alternative answer that was not identified in our interview study, we first checked whether it was similar to an existing choice. If not, we considered it as a unique answer and created a short, descriptive label for it. We then merged all the alternative answers with the same label and plotted their distributions in the histogram as well. For linear-scale questions, we calculated the total number and percentage of each option, and used the percentage to illustrate survey participants' evaluation of the importance of industrial needs, such as Figure 6. For open-ended questions, the first two authors used MAXQDA to code each answers individually, classified these answers according to sections in this work, and discussed with each other to refine codes and classifications.

3.3 Literature Review

We created the literature review protocol including research question, literature search strategy, literature selection criteria, literature selection procedures, data extraction strategy, and synthesis of the extracted data, following the methodology described in [68] to guide the literature review process. Our research question of literature review is to investigate whether existing research works related to ADS testing are adequate for the challenges and needs identified in our interviews and surveys. The literature search strategy is to search research papers from 28 SE conferences, journals and workshops including ICSE, ESEC/FSE, ASE, ISSTA, ISSRE, ICST, QRS, TSE, JSS, and IST. Specifically, we used 35 keywords to identify research papers related to autonomous driving, e.g., "autonomous driving", "autonomous vehicle", "traffic scene", "Apollo", etc. We built a crawler to download papers from publisher websites and found 117 papers that contain at least one of the keywords in their title or abstract. The literature selection criterion is to include technical papers that propose ADS testing methods and empirical papers that discuss challenges and solutions related to ADS testing. We manually reviewed all searched papers and found that 102 papers related to ADS, among which 42 are specifically about ADS testing[7]. We categorized these 42 ADS testing papers into different categories based on their research questions and solutions. The result is summarized in Section 6.1.

4 COMMON PRACTICES OF ADS TESTING

This section summarizes the common practices of ADS testing. Section 4.1 discusses the types of ADSs that our interview and survey participants worked on. Section 4.2, Section 4.3 and Section 4.4 discuss three commonly used ADS testing methods—*unit testing*, *real-world testing* and *simulation testing*. We do not separately present the results of the interview study and the survey for two main reasons. First, since the purpose of the survey is to confirm and supplement the qualitative findings from the interview, the findings from the interview study and the survey study have a lot of overlap. Therefore, if we report the findings separately, there will be a lot of redundancy. Second, fusing quantitative evidence (such as statistics from a large-scale survey) and qualitative evidence (such as quotations from interview participants) is more convenient for readers to read and understand the results of the survey.

4.1 Autonomous Driving Systems under Test

Our participants mainly work on two main types of ADSs: *multi-module driving systems* and *end-to-end driving models*.

Multi-module driving systems. The majority of interviewees (100%) and survey participants (69%) developed and tested multi-module driving systems. Multi-module architectures are widely used in industry-scale driving systems, e.g., Autoware [69] and Apollo [55]. They contain several modules for perception, prediction, planning, and control. The perception module takes a variety of sensor data as input, such as road images, point clouds, and GPS signals, to detect surrounding objects. The prediction module predicts the moving trajectories of these surrounding objects. Given the perception and prediction results, the planning module then decides on the route of the ego-vehicle. Finally, the control module converts the planned route to vehicle control commands, including braking force and steering angle. Four interview participants (P1, P3, P7, P9) elaborated that the perception and prediction modules in their systems heavily use deep neural networks, while the planning and control modules typically contain traditional logic-based programs. This is consistent with a previous study on Apollo [70].

End-to-end driving models. 31% of survey participants said they worked on end-to-end (E2E) driving models, while none of the interviewees worked on E2E driving models. E2E driving models such as PilotNet [71] and OpenPilot [72] treat the entire driving pipeline as a single deep learning model, from processing sensor data to generating vehicle controls. During the interview, participants pointed out that E2E driving models are less preferred in

[6] The survey result is publicly available here: https://bit.ly/38K9Csk

[7] The keyword list, venue list and paper list are publicly available at https://bit.ly/3FLUwCz

the industry, since they cannot handle complicated driving scenarios and often suffer from generalizability issues. For example, P7 said, *"Training an E2E driving model requires a large amount of data. And E2E driving models can easily over-fit and perform worse than multi-module systems."*

Common Practice 1

The majority of ADS practitioners reported to work on multi-module ADSs rather than end-to-end driving models. Therefore, multi-module ADSs deserve more attention in future research.

4.2 Unit testing

80% of interviewees and 52% of survey participants reported that they conducted unit testing during ADS development.

Testing target. In addition to writing unit tests for control logic, ADS developers also need to test DL models, especially those models in the perception and prediction modules. Unlike program source code, these models do not have clearly control logic or program states. ADS developers need to manually label and clip driving recordings collected from on-road or simulation testing to construct small recording segments for testing these DL models, which are referred to as unit tests in ADS development. There interviewees (P1, P4, P9) reported that there are too many possible driving scenarios to test, and it is time-consuming to manually process driving recordings to construct test scenarios.

In addition, 70% of interviewees and over 68% of survey participants said their driving systems used at least three of four types of sensors, e.g., cameras, LiDARs, radars, GPS. This multi-modality of sensor data makes it more difficult to generate test cases. For example, data from different sensors with different sample frequencies need to be synchronized (e.g., by timestamp). Furthermore, when transforming one type of sensor data, such as adding an object, other types of sensor data must be updated consistently.

Common Practice 2

In addition to testing control logic, ADS developers also need to construct segments of driving recordings to test DL models, which take multi-modal sensor data as input, not just road images.

Test metrics. When measuring the performance of a model, ADS developers use metrics, including accuracy, precision, recall, Receiver Operating Characteristic (ROC), and Intersection over Union (IoU). Furthermore, 51% of survey participants said they also use specific metrics tailored for different ADS modules. Two interview participants (P1, P9) elaborated on this—consistency is one of their metrics when testing the lane detection model. The lane detection results between the front and back frames in a video frame should be consistent.

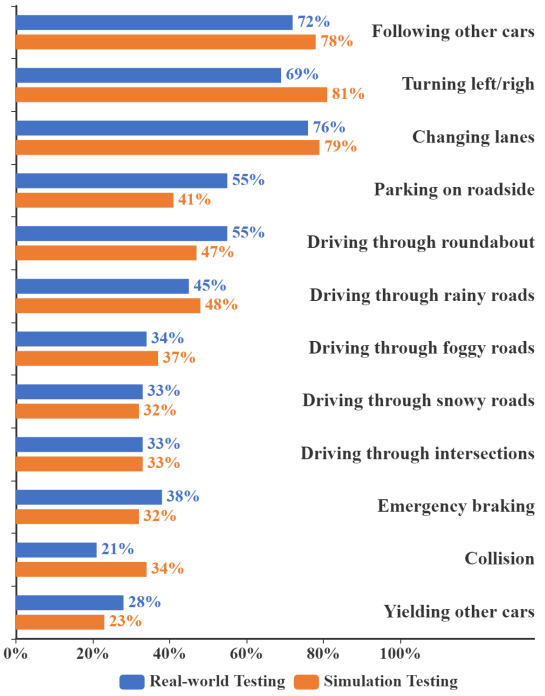

Figure 2: Common driving scenarios tested in on-road testing and simulation testing, respectively

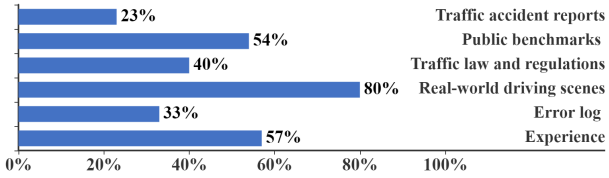

Figure 3: Sources of tested driving scenarios

Common Practice 3

ADS practitioners not only use common model performance metrics, such as accuracy and IoU, but also custom-tailored metrics such as consistency when testing perception models in ADS.

4.3 Real-world testing

All interview participants and 57% of survey participants reported that they have done real-world testing. Two types of real-world testing are mentioned: *scenario-based testing* and *on-road testing*.

Scenario-based testing. 4 interview participants (P1, P2, P8, P9) and 89% survey participants mentioned they have tested specific driving scenarios in closed automated vehicle proving grounds. Figure 2 shows commonly tested driving scenarios, as reported in our survey. It includes common driving scenarios such as changing lanes (76%), following other cars (72%), and turning left or right (69%), as well as special road sections such as intersections (33%) and roundabouts (over 55%). Weathers such as rainy days (44%)

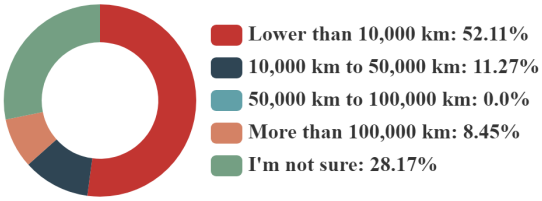

Figure 4: Length of on-road testing performed by industrial survey participants

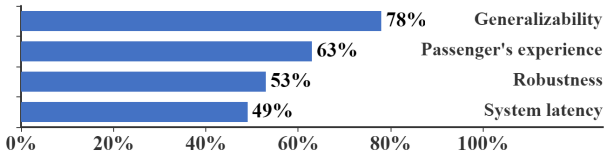

Figure 5: System-level metrics used in real-world testing

and snowy days (33%) are considered as well. Finally, dangerous driving scenarios such as emergency braking and collision are also conducted by 28% and 21% of survey participants, respectively. In the intervew study, P1 and P2 said they have built their own scenario database following traffic law and regulations. P8 and P9 said setting up those driving scenarios were time-consuming, which often took 2 weeks to 1 month.

Figure 3 shows the sources where those driving scenarios were from, as reported in the survey . 80% and 57% of survey participants said those scenarios were based on common real-world driving scenes and driving experiences. 54% of survey participants said they referred to public benchmarks such as CityScapes [73], ApolloScape [70], and Waymo open dataset [74]. 40% of survey participants said they referred to traffic laws and regulations. 23% said they referred to traffic accident reports.

Common Practice 4

ADS practitioners design various driving scenarios based on real-world driving scenes, public benchmarks, traffic laws and regulations, and crash reports to test an ADS in the field.

On-road testing. In on-road testing, autonomous vehicles are tested in public roads where various scenarios and unexpected conditions could occur. In practice, ADS companies need to carry out long-distance on-road testing to ensure the reliability of their driving systems. The number of kilometers an ADS travels without human intervention (i.e., km per disengagement) in on-road testing is used to measure the reliability of their ADS. As shown in the Disengagement from California Department of Motor Vehicles [75], Waymo conducted over 1 million kilometers of on-road testing in 2020, and its distance for each disengagement was about 48,000 km. During the interview, P8 said, "*the entire on-road testing required 50,000 to 100,000 kilometers, which must include different driving scenes such as highways, country roads, and urban roads.*" Furthermore, P9 mentioned that testers had to sit in the car and record driving data in on-road testing. The collected data would be either saved in local storage or uploaded to cloud platform, some of which would be cleaned and labelled for model training and testing. While on-road testing is critical, the reality is that, due to technical and financial constraints, many ADS practitioners can only conduct on-road testing within a limited range of mileage. Figure 4 shows that only 8% of participants conducted more than 100,000 km on-road testing, and 11% of them conducted 10,000 to 50,000 km on-road testing. As discussed in the next section, simulation testing is often considered as a more affordable and safer testing option for the majority of ADS practitioners.

Common Practice 5

On-road testing is considered as critical to ensure ADS reliability and robustness, while only a small portion of ADS practitioners have done long-distance testing.

Test metrics. Compared with metrics used in unit testing, more system-level metrics are used in real-world testing. As shown in Figure 5, four system-level metrics are frequently mentioned by survey participants, including generalizability (78%), passenger's experience (63%), robustness (53%), and system latency (49%). Generalizability focuses on whether an ADS can achieve similar performance in unseen driving scenes. Passenger's experience is another important metric in real-world testing. P4 said, "*when there are many vehicles in a driving scene, an ADS should not press the brake too frequently, which may make passengers uncomfortable.*" Robustness assesses whether an ADS can behave normally when noises, external interference, or attacks exist. As an ADS is expected to make quick, continuous responses to the change of driving scenes, system latency is often used to measure the decision-making speed of an ADS.

Common Practice 6

Test metrics used in real-world testing focus more on system-level performance, including both functional and non-functional properties, rather than only model accuracy.

4.4 Simulation testing

5 interviewees and 87% of survey participants said they conducted ADS testing in a simulation platform such as Carla [6], AirSim [76] and LGSVL [77]. When asked why simulation testing is so widely practiced, participants mentioned two main reasons. First, 54% of survey participants supported that modern simulation environments are powerful enough to test corner cases that are costly to set up in the real world. In the interview study, P2 and P6 mentioned that for dangerous driving scenarios such as collisions, simulation testing is much more preferred due to safety concerns. As shown in Figure 2, more participants did collision test in simulation than in the real world. Second, P2 and P9 mentioned that simulators can replay sensor data and vehicle control commands collected from real-world testing, which can be used to test the performance of a new release of an ADS. 55% of survey participants also support this point of view. Simulation testing is an important part of regression testing in ADS development. When developing an ADS, it is costly

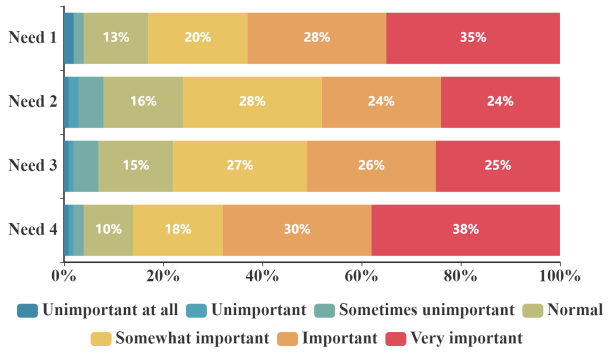

Figure 6: The importance of four common needs voted by survey participants. Details of each need are described in Section 5.

and time-consuming if a developer tests every commit of ADS in the real-world. Leveraging the simulation platform, industrial practitioners do not need to deploy each commit of ADS on the real vehicle and construct real-world test scenarios.

Common Practice 7
Simulation testing is widely adopted as a complement for real-world testing. It is particularly adopted to test new commits as part of regression testing.

Test metrics. Test metrics used in simulation testing is similar as on-road testing, including accuracy-based metrics, generalizability, passenger's experience, and robustness. However, as the simulation platform is built as an ideal environment without hardware delay, system latency is less considered in simulation testing.

5 EMERGING NEEDS

This section summarizes four emerging needs of ADS testing identified in the interview and survey. Figure 6 shows survey participants' agreement on the importance of these needs. 68%, 63%, 51% and 48% of survey participants considered Need 4, Need 1, Need 3, and Need 2 important or very important. Especially for Need 4 and Need 1, they are regarded as very important by more than one-third of survey participants.

5.1 Need 1: Identifying possible corner cases and unexpected driving scenarios

As shown in Figure 6, 35% and 28% of survey participants rated this need *very important* and *important* respectively. While modern driving systems are tested with diverse driving scenarios and extensive on-road testing, new corner cases are still often found during on-road testing. As P9 said, *"During on-road testing on California highways, we found it difficult to distinguish lane lines under the sunset. This was a problem we did not expect when we perform scenario-based testing."* Based on our conversation with ADS practitioners, the current industry practice to discover more corner cases seems to be simply performing more and longer on-road testing. However, unlike large automotive companies, small companies often lack resources (e.g., vehicle fleets) and on-road testing certificates to perform large-scale on-road testing. Therefore, identifying corner cases efficiently is an urgent need for ADS practitioners.

5.2 Need 2: Speeding up ADS testing

As shown in Figure 6, 24% and 24% of survey participants rated this need *very important* and *important* respectively. According to some existing work [63, 78], the catastrophic failure rate of an ADS should be minimized to 10^{-7} to 10^{-9} for 1 to 10 hours driving to achieve the goal of high reliability. To verify that the failure rate falls within one per 10^7 hours, one must conduct at least 10^7-hour testing of an ADS (about 1141 vehicle-years). It is unrealistic to achieve this goal with on-road testing. Therefore, ADS practitioners often resort to simulation testing. Widely-used simulators such as CarSim [79] support acceleration that only takes one-tenth of the time compared with real-world testing. However, given the large amount of time required to achieve high reliability (i.e., 10^7-hour driving test), it is still time consuming to conduct test in simulation. As P2 said, *"Our original goal was to do 20,000 kilometers of testing. Later, we found that even if it accelerates the speed by 5 times, the speed of the test is still slow, compared with our expectation."* Therefore, simulation acceleration is not a silver bullet to this challenge. Other methods to speed up ADS testing are needed. For example, test selection and prioritization approaches have been widely investigated in tradition software systems, which can be leveraged to accelerate ADS testing. We discuss this in detail in Section 6.3.

5.3 Need 3: Tool Support for Constructing Complex Driving Scenarios

As shown in Figure 6, 25% and 26% of survey participants rated this need *very important* and *important* respectively. ADS practitioners design driving scenarios based on various sources such as real-world driving scenes and traffic accident reports (Section 4.3). After identifying driving scenarios worth testing, they need to construct corresponding test cases. 65% of survey participants found it cumbersome to use toolkits (e.g., domain-specific languages, libraries, etc.) provided by existing simulation platforms to construct a test case of a driving scenario. Take OpenScenario [80] as an example. It takes 258 lines of code to construct a simple driving scenario of lane cutting, not even to mention complex scenarios. The main reason is that existing simulation platforms only provide low-level APIs and domain-specific languages (DSLs) to construct driving scenarios. While such low-level API and language design ensures the flexibility and expressiveness to precisely specify arbitrary driving scenarios, it imposes significant coding effort. Similar to how Google releases Keras as a higher-level abstraction of TensorFlow, it would be beneficial to provide a higher-level abstraction of the APIs and DSLs in the simulation platforms. In addition, as mentioned by P8, "*We already collected traffic accident video from internet, but it is still hard to automatically transform them into test cases in simulators.*" ADS developers wish they can get more tool support that helps them automatically or semi-automatically construct driving scenarios in a simulation environment, such as translating a natural language description of a traffic scene into the low-level code written in APIs provided by a simulation environment.

5.4 Need 4: Tool support for data labeling

As shown in Figure 6, 38% and 30% of survey participants rated this need *very important* and *important* respectively. Section 4.3 has already discussed that driving data collected in on-road testing, such as point clouds and road images, are often used as test data. However, as mentioned by P8, *these driving data need to be manually labelled and clipped first before they can be replayed in a simulation environment or reused to test a DL model.* Two typical labels are 2D bounding boxes and semantic segmentation masks. A 2D bounding box label includes the height and width of a detected object and the type of the object. And a semantic segmentation mask requires data labelers to manually segment all objects at pixel level. Given the massive amount of driving data collected from on-road testing, the labeling effort is enormous. While there have been some assistive labeling tools in recent years [81–83], labeling driving data still requires many manual effort. For example, Amazon SageMaker [81] can automatically assign labels for object detection and image segmentation tasks. But it requires data labelers to manually go through those labels and make corrections. Several ADS companies we have interviewed said they often outsourced the data labeling task to data labeling companies or used crowdsourcing platforms such as Amazon Mechanic Turks [84]. However, even for manually labeled data, label quality is still a concern, especially for safety-critical tasks like autonomous driving. According to a recent study [85], several well-known datasets such as ImageNet are riddled with manual labeling mistakes. Therefore, ADS practitioners wish to get more tool support to analyze, recognize, and repair labeling errors in their driving data.

6 LITERATURE REVIEW AND RESEARCH GAPS

The previous section summarizes four emergent needs from industrial practitioners in ADS testing based on interviews and surveys. To understand to what extent state-of-the-art ADS testing techniques have addressed these needs, we surveyed research papers from software engineering conferences and journals and manually assessed them. This section summarizes our findings and future research opportunities.

6.1 Literature Taxonomy

Figure 7 describes our taxonomy of ADS testing techniques, based on the 42 papers identified from Section 3.3.

Corner case generation. A variety of techniques have been proposed to generate corner cases for ADSs. We categorized them into two lines of research. One line of research is *knowledge-based methods* [22–32]. Knowledge-based methods generate corner cases using domain-specific ontology or metamorphic relations, which are designed based on human driving experience, traffic accident reports, traffic law and regulations. For example, DeepTest [23] and DeepRoad [22] generated corner cases using metamorphic relations from common sense that weather changes should not affect the steering angle prediction of an ADS. AC3R [27] used domain-specific ontology and natural language processing to extract information from police crash reports and reconstruct corner cases of crash accidents. DeepBillboard [32] generated corner cases by adding adversarial perturbation [86] patches on real-world billboard to check whether the E2E driving model keeps the same output of the steering angle.

Search-based method [7–21] is another line of research in corner case generation for ADSs. These methods aim to find a set of test parameters that introduce behavior discrepancies in an ADS, such as collisions and lane departures. To define a parameter search space, the majority of search-based methods leverage *driving patterns* [7–19]. Driving patterns are designed based on the states of an autonomous vehicle, such as its position and speed, and other vehicles and pedestrians on the road. The search process is normally guided by manually defined fitness functions such as time-to-collision and the minimal distance between the ego-vehicle and pedestrians. Optimal solutions of such fitness functions are found by applying genetic algorithms [7, 14, 15, 19]. For example, AsFault [7] used genetic algorithm to evaluate parameters of road, like length and position, to make the ego-vehicle more error-prone on lane keeping task on generated test cases. Abdessalem et al. [20] proposed a search algorithm that combines the genetic algorithm with a decision tree classifier to guide the test case generation faster towards critical test scenarios. Gladisch et al. used Bayesian optimization as an alternative for genetic algorithms [21].

Test selection and prioritization. We identified 5 papers [17–19, 33, 34] on this research direction. These methods select or prioritize test cases according to the similarity between the vector representation of each test case. Kim et al. [33] encoded a test case as an embedding vector using the output of a set of selected neuron outputs, and used the likelihood-based surprise adequacy and Mahalanobis-distance-based surprise adequacy to measure the difference between a test case with others by estimating the density of vector representations of each selected test case. Test cases with high surprise adequacy would be selected as critical test cases.

Empirical Studies. We found 4 empirical studies on ADS testing. Knauss et al. [38] conducted an empirical study including focus groups and interviews to identify challenges in ADS testing. Jahangirova et al. [37] performed a systematic analysis on metrics of driving qualities, and used a mutation analysis to select 26 metrics, such as the max lateral position and the standard error of speed, as functional oracles for ADS testing. These selected oracles can kill mutants of the E2E driving model with low false alarm rate. Liu et al. [36] analyzed autonomous vehicle disengagement and collision reports from the California Department of Motor Vehicles [4] and found that the growth of on-road testing mileage is not accompanied by the increase of the disengagement ratio. Garcia et al. [35] investigated on commits and ADS bugs in Apollo [55] and Autoware [56], and classified found bugs into 13 root causes, such as incorrect algorithm implementations, incorrect condition logic and concurrency.

Simulation Testing. 4 SE papers have discussed simulation testing. Hu et al. [40] and Masuda et al. [43] discussed the potential issues in simulation testing, including huge input space and precision issues in the simulator. Haq et al. [42] discussed the reliability of driving data generated by simulators. Leudet et al. [41] designed a simulator called AILiveSim, which is not only focused on autonomous vehicles, but can also be used to simulate autonomous ships and autonomous mining machines.

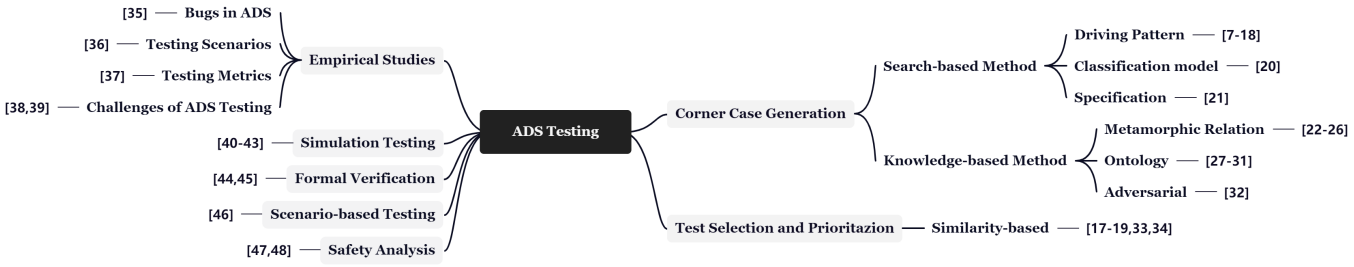

Figure 7: A taxonomy of ADS testing research.

Formal Verification. Fritzsch et al. [44] reported their experience of employing bounded model checking on a symbolic model checker NuSMV to verify relevant properties for a vehicle control system. Du et al. [45] used a DSL to specify relationships among driving scenario elements and used Stochastic Hybrid Automata to specify the dynamic behaviors, which can then be checked by an existing model checker UPPAAL-SMC. The approach can be used to verify safety-related properties of a specific driving scenario.

Scenario-based testing. King et al. [46] proposed a scenario-based testing method to test the adaptive cruise control function in ADS. The proposed method can parallel evaluate multiple systems in multiple similar but unknown test scenarios. And the prototype is

Safety Analysis. Salay et al. [47] introduced a classification-specific safety analysis tool based on failure mode effects analysis. The method named CFMEA is able to identify failure modes and risk levels in perception models of ADS. Similarly Zhao et al. [48] used a variant of Conservative Baysian inference to avoid catastrophic failure due to overconfidence for a few inference tools used in AV.

6.2 Gaps and Future Directions for Need 1

Among the four emerging needs identified in our study, the need of identifying corner cases and unexpected driving scenarios is the one that receives the most attention from the research community. As discussed in Section 6.1, a large number of corner case generation techniques have been proposed for ADSs. These techniques have already constituted of good solutions that we believe ADS practitioners should try out. Here we summarize several improvement opportunities for the existing techniques. First, for ontology-based or metamorphic relation-based methods, it is worth investigating the generation of more complex driving scenarios (e.g., lane merging, overtaking) beyond, for instance, affine transformations and weather conditions. Second, for search-based methods, it is always challenging to define a tractable search space and fitness function for realistic, complex driving scenarios. As a future direction, it is worth investigating how to elicit new safety requirements from existing crash reports, traffic rules and on-road driving recordings. Such requirements can be used to generate new fitness functions or refine existing ones to augment the search space for more mission-critical testing scenarios.

Furthermore, as discussed in Section 4.1, all interview participants and 69% of survey participants are now working on multi-module ADSs. These multi-module ADSs take multiple types of sensor data as input. Yet the majority of test generation techniques only generate road image data. Therefore, it is worth investigating how to generate multi-modal sensor data for new driving scenarios.

6.3 Gaps and Future Directions for Need 2

Five papers on test selection and prioritization can be applied to address the need of accelerating ADS testing [17–19, 33, 34]. These techniques can be used to remove redundant testing scenarios and prioritize driving recordings that are more likely to expose errors in an ADS. For example, when driving on a highway, the ego-vehicle often goes straight on a specific lane at a constant speed. Therefore, there is no need to replay the entire highway driving recording but only unique driving scenarios during the recording, such as lane cutting. All these technical papers use similarity-based test metrics. Researchers can also investigate other kinds of metrics, such as confidence-based metrics [87–90], to select or prioritize test cases. For example, DeepGini [87] applied Gini impurity on confidence to measure the likelihood of misclassification on an input, and prioritized test cases with high Gini scores. Furthermore, instead of using test metrics, we also suggest that researchers consider dynamic HD maps, which is an intermediate representation of driving scenes used by modern ADSs. HD maps incorporate the outputs of perception models and are then fed as input for the following prediction and control modules. They can be used as a holistic representation for driving scenes.

Since current ADS test prioritization techniques focus on end-to-end (E2E) driving models or only the path planning module in an ADS, there is an opportunity to develop new techniques for multi-module ADSs. To account for multi-module architectures, researchers may want to investigate how to combine test metrics for inner model behaviors, such as neuron coverage and surprise adequacy, and logic-based test metrics, such as path coverage, to capture the interaction among multiple models, and interaction between models and logic-based control modules. One possible solution is to formulate this as a multi-objective optimization problem. A technique should search for a minimal set of test cases that collectively maximize the coverage of individual ML models as well as the path or dependency coverage of an ADS. For example, a driving scene at a busy traffic intersection is likely to have higher collective coverage at the model and system levels compared with a highway scene with little traffic, since it involves multiple types of objects (e.g., pedestrians, vehicles, and traffic lights) that may trigger multiple execution paths in an ADS and also trigger multiple perception models. Given the large number of possible test case combinations, it is worth investigating how to leverage

multi-objective optimization algorithms to efficiently search for an optimal test set.

6.4 Gaps and Future Directions for Need 3

We have only identified two relevant SE papers [27, 45] that addressed the need of constructing complex driving scenarios. Gambi et al. [27] proposed an automated approach called AC3R that parses crash reports to driving scenarios in a simulation environment. Specifically, AC3R leverages NLP techniques to extract key information (e.g., traffic participants and driving actions) from crash reports and transform crash information to a Lua script to control traffic participants and generate corresponding crash scenarios. AC3R is limited to some simple crash scenarios like rear-end collision of two vehicles. In addition, as crash reports often follow rigid narrative standards, AC3R may have a difficult time translating free-form driving scene descriptions to an executable driving scene. As future work, it is worth investigating new techniques in NLP and CV to support automated construction of driving scenes from text or video data. Du et al. [45] proposed a new modeling language that describes specifications in various driving scenarios for the purpose of formal verification. While this new language is not directly used to construct new test cases, it can be adapted as a domain-specific language (DSL) for test generation.

We also searched more broadly online and found some effort in addressing this need from other research communities. The Intelligent Vehicle and Programming Language communities has proposed several higher-level DSLs to reduce the effort of modeling objects and motions in a simulation platform [91–94]. For instance, Scenic [91] is a probabilistic programming language which enables developers to generate complex traffic jam scenarios with just a few lines of code. These newly designed DSLs are more compact and flexible. We believe SE researchers can also contribute to this line of research.

6.5 Gaps and Future Directions for Need 4

Among all four emerging needs, the need of tool support for data labeling receives the highest importance rating in the survey study. 68% of survey participants considered this need as very important or important. We have only identified one SE paper [33] that attempts to address this need. Kim et al. [33] proposed to surprise adequacy, a simple metric to quantify how surprising an input is to a DNN, to guide the selection of new road images for labeling. The result is promising — 30 to 50% of manual labeling cost can be saved with negligible impact on evaluation accuracy. However, reducing the amount of labels does not help improve label quality. Despite less effort in this direction so far, we have noticed that there is an increasing attention to the general problem of data labeling in the SE community. A case study by Amershi et al. [61] recognized data collection, cleaning, and labeling as crucial steps in the development pipeline of machine learning applications. Other communities such as database and computer vision have proposed several supervised or semi-supervised learning techniques to automatically label data [95–99]. We suggest that readers refer to Roh et al. [100] for a detailed survey of data collection and labeling tools built by other research communities. In the future, we believe it is worthwhile developing more automated or semi-automated tools such as [33] to support reducing data labeling efforts in ADSs.

Furthermore, SE researchers can also contribute to data cleaning by developing input validation or outlier detection techniques on driving data. Traditional data cleaning approaches need users to provide a set of specific rules to determine constraints [101–103]. Such rules are easy to define on tabular data or text data. However, it is challenging to define such rules for sensor data such as videos and point clouds that involve complex driving scenes. The SE community has proposed many techniques to identify failure-inducing inputs in software testing and debugging [104–109]. For example, Gulzar et al. [107] leveraged data provenance to trace error propagation in big data systems and identify fault-inducing data. It would be beneficial to investigate how to adapt or renovate these techniques to identify low-quality data that leads to abnormal driving behavior in ADSs.

7 THREATS TO VALIDITY

We discuss the threats to validity following the standard proposed by Wohlin et al. [110].

External Validity. The external validity concerns about the generalizability of our findings. To mitigate the threat to external validity, we improved the diversity of the interviewees as much as possible. As shown in Section 2.1, these 10 interviewees came from 10 different companies and had different responsibilities. Furthermore, to expand the scale of our research, we also conducted 100 surveys, and these respondents were in a variety of positions, and came from various types of companies and organizations, as shown in Section 2.2. In addition, ADS developed and tested by these interviewees and questionnaire respondents were from level 1 to level 5 [111], which covered all levels of industrial ADS. Therefore, the interviewees and survey respondents in this study can represent the industrial participants of ADS testing.

Construct Validity. The construct validity concerns about the design of our empirical study and literature review. To mitigate this threat, for interviews, we first conducted two pilot interviews. Based on feedback from these two pilot interviews, we refined the interview guide and interview questions. In addition, to not bias participants' responses during the interview, we conducted semi-structured interviews and asked open-ended questions, which allowed for new ideas and questions to be brought up during the discussion. Though the survey questions were mainly multiple-choice questions, we allowed respondents to provide alternative answers. Also open-ended questions were added in the survey for respondents to supplement questions and answers not included in the survey. In the literature review, we used broad keywords to include as many relevant papers as possible. Additionally, with manual filtering, we removed papers that were not relevant to autonomous driving testing. Though in the literature review process, we did not apply literature quality assessment as described in [68], the quality of searched papers can be assured because they are all published on high quality SE conferences and journals. We did not create detailed data collection forms for searched papers, which might affect the categorization of reviewed papers. The process of data collection will be refined in the future work.

Internal Validity. The internal validity concerns about how accurate participants' responses may reflect their real needs and whether there are other compounding factors that may affect the results. Specifically, since interview participants were developers from technology companies, some of them may not faithfully reveal the exact and most emergent needs from their team due to confidentiality concerns. When designing the interview, we tried our best to avoid conversations that may involve technical details or make them feel uncomfortable to answer. In addition, before each interview, we shared the interview guide with the participant and informed them that our goal was to solicit high-level feedback rather than technical details. To avoid mistakes and biases in the thematic analysis step, the first and second authors carried out this step separately. Then two authors continuously adjusted the extraction result, until the inter-rater agreements between the two coders reached 0.85, in Cohen's Kappa [66]. Also, the collected information was regularly reported and discussed in the whole research team.

8 CONCLUSION

This paper presents an empirical study on the current practices and needs of testing autonomous driving systems. Through an interview study and a large-scale survey, we identified seven common practices and four common needs of ADS practitioners. These findings provide a deep understanding about how ADS practitioners test autonomous driving systems in practice and what kinds of tool support they find helpful. Furthermore, we did a literature review of related ADS testing research from the SE community and assessed to what extent existing work can address those industrial needs. We found that, though one need, *identifying possible corner cases and unexpected driving scenarios*, is widely investigated by the SE community, the other three needs are under-investigated. Furthermore, most existing work is designed for end-to-end driving models, while multi-module driving systems are widely adopted nowadays. Based on these gaps, we proposed four future directions to build better tool support for testing autonomous driving systems. We believe there are many research opportunities for SE researchers and therefore call for more attention and effort towards these future directions.

9 ACKNOWLEDGEMENTS

This work is in part supported by an Australian Research Council (ARC) Discovery Project (DP210102447), an ARC Linkage Project (LP190100676), and a DATA61 project (Data61 CRP C020996).

REFERENCES

[1] S. Gibbs. Google sibling waymo launches fully autonomous ride-hailing service. *The Guardian*, 7, 2017.
[2] Arthur Frederick Hasler. 60,000 drivers now have tesla full self driving (fsd)—what it is how to get it. https://bit.ly/3t1zIST, 2022.
[3] The National Transportation Safety Board. Preliminary report highway hwy18mh010. https://bit.ly/2N0SHuj. Retrieved: 2019.
[4] California DMV. Autonomous vehicle collision reports - california dmv. https://bit.ly/3cPUcGC. Retrieved: 2019.
[5] F. Favar, S. Eurich, and N. Nader. Autonomous vehicles' disengagements: Trends, triggers, and regulatory limitations. *Accident Analysis & Prevention*, 110:136–148, 2018.
[6] Carla. Carla: Open-source simulator for autonomous driving research. https://bit.ly/3qE26qA, 2021.
[7] Alessio Gambi, Marc Müller, and Gordon Fraser. Asfault: Testing self-driving car software using search-based procedural content generation. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 27–30. IEEE, 2019.
[8] Yun Tang, Yuan Zhou, Tianwei Zhang, Fenghua Wu, Yang Liu, and Gang Wang. Systematic testing of autonomous driving systems using map topology-based scenario classification. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1342–1346. IEEE, 2021.
[9] Guanpeng Li, Yiran Li, Saurabh Jha, Timothy Tsai, Michael Sullivan, Siva Kumar Sastry Hari, Zbigniew Kalbarczyk, and Ravishankar Iyer. Av-fuzzer: Finding safety violations in autonomous driving systems. In *ISSRE*, pages 25–36. IEEE, 2020.
[10] Fuyuki Ishikawa. Testing and debugging autonomous driving: Experiences with path planner and future challenges. In *2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages xxxiii–xxxiv. IEEE, 2020.
[11] Paolo Arcaini, Xiao-Yi Zhang, and Fuyuki Ishikawa. Targeting patterns of driving characteristics in testing autonomous driving systems. In *ICST*, pages 295–305. IEEE, 2021.
[12] Markus Borg, Raja Ben Abdessalem, Shiva Nejati, François-Xavier Jegeden, and Donghwan Shin. Digital twins are not monozygotic–cross-replicating adas testing in two industry-grade automotive simulators. In *ICST*, pages 383–393. IEEE, 2021.
[13] Alessandro Calò, Paolo Arcaini, Shaukat Ali, Florian Hauer, and Fuyuki Ishikawa. Generating avoidable collision scenarios for testing autonomous driving systems. In *ICST*, pages 375–386. IEEE, 2020.
[14] Raja Ben Abdessalem, Shiva Nejati, Lionel C Briand, and Thomas Stifter. Testing vision-based control systems using learnable evolutionary algorithms. In *ICSE*, pages 1016–1026. IEEE, 2018.
[15] Alessio Gambi, Marc Mueller, and Gordon Fraser. Automatically testing self-driving cars with search-based procedural content generation. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 318–328, 2019.
[16] F. Kluck, M. Zimmermann, F. Wotawa, and M. Nica. Genetic algorithm-based test parameter optimization for ADAS system testing. In *QRS*. IEEE, jul 2019.
[17] Galen E Mullins, Paul G Stankiewicz, R Chad Hawthorne, and Satyandra K Gupta. Adaptive generation of challenging scenarios for testing and evaluation of autonomous vehicles. *Journal of Systems and Software*, 137:197–215, 2018.
[18] Chengjie Lu, Huihui Zhang, Tao Yue, and Shaukat Ali. Search-based selection and prioritization of test scenarios for autonomous driving systems. In *International Symposium on Search Based Software Engineering*, pages 41–55. Springer, 2021.
[19] Yixing Luo, Xiao-Yi Zhang, Paolo Arcaini, Zhi Jin, Haiyan Zhao, Fuyuki Ishikawa, Rongxin Wu, and Tao Xie. Targeting requirements violations of autonomous driving systems by dynamic evolutionary search. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 279–291. IEEE, 2021.
[20] Raja Ben Abdessalem, Annibale Panichella, Shiva Nejati, Lionel C Briand, and Thomas Stifter. Testing autonomous cars for feature interaction failures using many-objective search. In *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 143–154. IEEE, 2018.
[21] Christoph Gladisch, Thomas Heinz, Christian Heinzemann, Jens Oehlerking, Anne von Vietinghoff, and Tim Pfitzer. Experience paper: Search-based testing in automated driving control applications. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 26–37. IEEE, 2019.
[22] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. In *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 132–142. IEEE, 2018.
[23] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*, pages 303–314, 2018.
[24] Pablo Valle. Metamorphic testing of autonomous vehicles: a case study on simulink. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 105–107. IEEE, 2021.
[25] J. Han and Z. Zhou. Metamorphic fuzz testing of autonomous vehicles. In *ICSE Workshop*, pages 380–385. ACM, 2020.
[26] Jaganmohan Chandrasekaran, Yu Lei, Raghu Kacker, and D Richard Kuhn. A combinatorial approach to testing deep neural network-based autonomous driving systems. In *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 57–66. IEEE, 2021.
[27] Alessio Gambi, Tri Huynh, and Gordon Fraser. Generating effective test cases for self-driving cars from police reports. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 257–267, 2019.
[28] Florian Klück, Yihao Li, Mihai Nica, Jianbo Tao, and Franz Wotawa. Using ontologies for test suites generation for automated and autonomous driving functions. In *2018 IEEE International symposium on software reliability engineering workshops (ISSREW)*, pages 118–123. IEEE, 2018.

[29] Jianbo Tao, Yihao Li, Franz Wotawa, Hermann Felbinger, and Mihai Nica. On the industrial application of combinatorial testing for autonomous driving functions. In *2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 234–240. IEEE, 2019.

[30] Alessio Gambi, Tri Huynh, and Gordon Fraser. Automatically reconstructing car crashes from police reports for testing self-driving cars. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 290–291. IEEE, 2019.

[31] Yihao Li, Jianbo Tao, and Franz Wotawa. Ontology-based test generation for automated and autonomous driving functions. *Information and software technology*, 117:106200, 2020.

[32] Husheng Zhou, Wei Li, Zelun Kong, Junfeng Guo, Yuqun Zhang, Bei Yu, Lingming Zhang, and Cong Liu. Deepbillboard: Systematic physical-world testing of autonomous driving systems. In *ICSE*, pages 347–358. IEEE, 2020.

[33] Jinhan Kim, Jeongil Ju, Robert Feldt, and Shin Yoo. Reducing dnn labelling cost using surprise adequacy: An industrial case study for autonomous driving. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1466–1476, 2020.

[34] Christian Wolschke, Thomas Kuhn, Dieter Rombach, and Peter Liggesmeyer. Observation based creation of minimal test suites for autonomous vehicles. In *2017 IEEE International symposium on software reliability engineering workshops (ISSREW)*, pages 294–301. IEEE, 2017.

[35] J. Garcia, Y. Feng, J. Shen, S. Almanee, Y. Xia, Chen, and Q. Alfred. A comprehensive study of autonomous vehicle bugs. In *ICSE*, pages 385–396, 2020.

[36] Siyuan Liu and Luiz Fernando Capretz. An analysis of testing scenarios for automated driving systems. In *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 622–629. IEEE, 2021.

[37] Gunel Jahangirova, Andrea Stocco, and Paolo Tonella. Quality metrics and oracles for autonomous vehicles testing. In *ICST*, pages 194–204. IEEE, 2021.

[38] Alessia Knauss, Jan Schroder, Christian Berger, and Henrik Eriksson. Software-related challenges of testing automated vehicles. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 328–330. IEEE, 2017.

[39] Xudong Zhang, Yan Cai, and Zijiang Yang. A study on testing autonomous driving systems. In *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 241–244, 2020.

[40] Zhisheng Hu, Shengjian Guo, Zhenyu Zhong, and Kang Li. Disclosing the fragility problem of virtual safety testing for autonomous driving systems. In *2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 387–392. IEEE, 2021.

[41] Jérôme Leudet, François Christophe, Tommi Mikkonen, and Tomi Männistö. Ailivesim: An extensible virtual environment for training autonomous vehicles. In *2019 IEEE 43rd annual computer software and applications conference (COMPSAC)*, volume 1, pages 479–488. IEEE, 2019.

[42] Fitash Ul Haq, Donghwan Shin, Shiva Nejati, and Lionel C Briand. Comparing offline and online testing of deep neural networks: An autonomous car case study. In *ICST*, pages 85–95. IEEE, 2020.

[43] Satoshi Masuda. Software testing design techniques used in automated vehicle simulations. In *2017 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 300–303. IEEE, 2017.

[44] Jonas Fritzsch, Tobias Schmid, and Stefan Wagner. Experiences from large-scale model checking: Verifying a vehicle control system with nusmv. In *ICST*, pages 372–382. IEEE, 2021.

[45] Dehui Du, Jiena Chen, Mingzhuo Zhang, and Mingjun Ma. Towards verified safety-critical autonomous driving scenario with adsml. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1333–1338. IEEE, 2021.

[46] Christian King, Lennart Ries, Christopher Kober, Christoph Wohlfahrt, and Eric Sax. Automated function assessment in driving scenarios. In *ICST*, pages 414–419. IEEE, 2019.

[47] Rick Salay, Matt Angus, and Krzysztof Czarnecki. A safety analysis method for perceptual components in automated driving. In *ISSRE*, pages 24–34. IEEE, 2019.

[48] Xingyu Zhao, Valentin Robu, David Flynn, Kizito Salako, and Lorenzo Strigini. Assessing the safety and reliability of autonomous vehicles from road testing. In *ISSRE*, pages 13–23. IEEE, 2019.

[49] R Burke Johnson and Anthony J Onwuegbuzie. Mixed methods research: A research paradigm whose time has come. *Educational researcher*, 33(7):14–26, 2004.

[50] J. Zhang, M. Harman, L. Ma, and Y. Liu. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, 2020.

[51] W. Xiang, P. Musau, A. Wild, D. M. Lopez, N. Hamilton, X. Yang, J. Rosenfeld, and T. Johnson. Verification for machine learning, autonomy, and neural networks survey. *arXiv preprint arXiv:1810.01989*, 2018.

[52] S. Dey and S. Lee. Multilayered review of safety approaches for machine learning-based systems in the days of AI. *Journal of Systems and Software*, 176:110941, jun 2021.

[53] M. Borg, C. Englund, K. Wnuk, B. Duran, C. Levandowski, S. Gao, Y. Tan, H. Kaijser, H. Lönn, and J. Törnqvist. Safely entering the deep: A review of verification and validation for machine learning and a challenge elicitation in the automotive industry. *arXiv preprint arXiv:1812.05389*, 2018.

[54] Z. Peng, J. Yang, T. H. Chen, and L. Ma. A first look at the integration of machine learning models in complex autonomous driving systems: a case study on apollo. In *FSE*, pages 1240–1250. ACM, 2020.

[55] ApolloAuto. Apollo. https://bit.ly/2E3vWyo, 2021.

[56] Autoware-AI. autoware.ai. https://bit.ly/3gZ1gBS, 2020.

[57] T. Zhang, C. Gao, L. Ma, M. Lyu, and M. Kim. An empirical study of common challenges in developing deep learning applications. In *ISSRE*, pages 104–115. IEEE, 2019.

[58] Y. Zhang, Y. Chen, S. Cheung, Y. Xiong, and L. Zhang. An empirical study on tensorflow program bugs. In *ISSTA*, pages 129–140, 2018.

[59] F. Thung, S. Wang, D. Lo, and L. Jiang. An empirical study of bugs in machine learning systems. In *ISSRE*, pages 271–280. IEEE, 2012.

[60] X. Sun, T. Zhou, G. Li, J. Hu, H. Yang, and B. Li. An empirical study on real bugs for machine learning programs. In *Asia-Pacific Software Engineering Conference*, pages 348–357. IEEE, 2017.

[61] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann. Software engineering for machine learning: A case study. In *ICSE*, pages 291–300. IEEE, 2019.

[62] W. Huang, K. Wang, Y. Lv, and F. Zhu. Autonomous vehicles testing methods review. In *ITSC*, pages 163–168. IEEE, 2016.

[63] P. Koopman and M. Wagner. Challenges in autonomous vehicle testing and validation. *SAE International Journal of Transportation Safety*, 4(1):15–24, 2016.

[64] Forrest Shull, Janice Singer, and Dag I. K. Sjøberg, editors. *Guide to Advanced Empirical Software Engineering*. Springer, 2008.

[65] V. Braun and V. Clarke. Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2):77–101, 2006.

[66] A. Viera, J. Garrett, et al. Understanding interobserver agreement: the kappa statistic. *Fam med*, 37(5):360–363, 2005.

[67] DeepDrive. Deepdrive. https://bit.ly/2OTsheJ, 2020.

[68] Staffs Keele et al. Guidelines for performing systematic literature reviews in software engineering. Technical report, Technical report, ver. 2.3 ebse technical report. ebse, 2007.

[69] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi. Autoware on board: Enabling autonomous vehicles with embedded systems. In *ICCPS*, pages 287–296. IEEE, 2018.

[70] P. Wang, X. Huang, X. Cheng, D. Zhou, Q. Geng, and R. Yang. The apolloscape open dataset for autonomous driving and its application. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[71] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, Lawrence D. Jackel, and U. Muller. Explaining how a deep neural network trained with end-to-end learning steers a car. *CoRR*, abs/1704.07911, 2017.

[72] comma. ai. Openpilot. https://bit.ly/3w099fl, 2021.

[73] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

[74] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, pages 2446–2454, 2020.

[75] California Department of Motor Vehicles. Disengagement report. https://bit.ly/2NmXA1c. Accessed: 2021-02-20.

[76] microsoft. Airsim. https://bit.ly/3qREI8Q, 2021.

[77] lgsvl. simulator. https://bit.ly/3dBDif1, 2020.

[78] R. Butler and G. Finelli. The infeasibility of experimental quantification of life-critical software reliability. In *Conference on Software for Citical Systems*, pages 66–76, 1991.

[79] R. Benekohal and J. Treiterer. Carsim: Car-following model for simulation of traffic in normal and stop-and-go conditions. *Transportation research record*, 1194:99–111, 1988.

[80] ASAM. Asam openscenario. https://bit.ly/3ya34Rm. Accessed: 2021-02-20.

[81] A. Joshi. Amazon's machine learning toolkit: Sagemaker. In *Machine Learning and Artificial Intelligence*, pages 233–243. Springer, 2020.

[82] microsoft. Uvott. https://bit.ly/3rEWxJO, 2021.

[83] M. Sharma, D. Rasmuson, B. Rieger, D. Kjelkerud, et al. Labelbox: The best way to create and manage training data. software, labelbox. *Inc, https://bit.ly/2TBLzYW*, 2019.

[84] G. Paolacci, Ch, and J. ler. Inside the turk: Understanding mechanical turk as a participant pool. *Current directions in psychological science*, 23(3):184–188, 2014.

[85] C. Northcutt, A. Athalye, and J. Mueller. Pervasive label errors in test sets destabilize machine learning benchmarks. *arXiv preprint arXiv:2103.14749*, 2021.

[86] Yao Deng, Xi Zheng, Tianyi Zhang, Chen Chen, Guannan Lou, and Miryung Kim. An analysis of adversarial attacks and defenses on autonomous driving models. In *2020 IEEE international conference on pervasive computing and communications (PerCom)*, pages 1–10. IEEE, 2020.

[87] Y. Feng, Q. Shi, X. Gao, J. Wan, C. Fang, and Z. Chen. Deepgini: prioritizing massive tests to enhance the robustness of deep neural networks. In *ISSTA*, pages 177–188. ACM, 2020.

[88] W. Ma, M. Papadakis, A. Tsakmalis, M. Cordy, and Y. Traon. Test selection for deep learning systems. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 30(2):1–22, 2021.

[89] Z. Li, L. Zhang, J. Yan, J. Zhang, Z. Zhang, and T. H. Tse. PEACEPACT: Prioritizing examples to accelerate perturbation-based adversary generation for DNN classification testing. In *QRS*. IEEE, dec 2020.

[90] Z. Wang, H. You, J. Chen, Y. Zhang, X. Dong, and W. Zhang. Prioritizing test inputs for deep neural networks via mutation analysis. In *ICSE*, pages 397–409. IEEE, 2021.

[91] D. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. Sangiovanni-Vincentelli, and S. Seshia. Scenic: a language for scenario specification and scene generation. In *PLDI*, pages 63–78, 2019.

[92] R. Queiroz, T. Berger, and K. Czarnecki. Geoscenario: An open dsl for autonomous driving scenario representation. In *IV*, pages 287–294. IEEE, 2019.

[93] M. Althoff, M. Koschi, and S. Manzinger. Commonroad: Composable benchmarks for motion planning on roads. In *IV*, pages 719–726. IEEE, 2017.

[94] P. Bender, J. Ziegler, and C. Stiller. Lanelets: Efficient map representation for autonomous driving. In *IV*, pages 420–425. IEEE, 2014.

[95] B. Benato. Semi-automatic data annotation guided by feature space projection. *Pattern Recognition*, 109:107612, 2021.

[96] X. Ke, J. Zou, and Y. Niu. End-to-end automatic image annotation based on deep cnn and multi-label data augmentation. *IEEE Transactions on Multimedia*, 21(8):2093–2106, 2019.

[97] L. Chen, S. Fidler, A. Yuille, and R. Urtasun. Beat the mturkers: Automatic image labeling from weak 3d supervision. In *CVPR*, pages 3198–3205, 2014.

[98] B. Wu, W. Chen, P. Sun, W. Liu, B. Ghanem, and S. Lyu. Tagging like humans: Diverse and distinct image annotation. In *CVPR*, pages 7967–7975, 2018.

[99] X. Ke, M. Zhou, Y. Niu, and W. Guo. Data equilibrium based automatic image annotation by fusing deep model and semantic propagation. *Pattern Recognition*, 71:60–77, 2017.

[100] Y. Roh, G. Heo, and S. Whang. A survey on data collection for machine learning: a big data-ai integration perspective. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[101] V. Gudivada, A. Apon, and J. Ding. Data quality considerations for big data and machine learning: Going beyond data cleaning and transformations. *International Journal on Advances in Software*, 10(1):1–20, 2017.

[102] X. Chu, I. Ilyas, S. Krishnan, and J. Wang. Data cleaning: Overview and emerging challenges. In *International Conference on Management of Data*, pages 2201–2206, 2016.

[103] E. Rahm and H. Do. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13, 2000.

[104] A. Zeller, Hildebr, and R. t. Simplifying and isolating failure-inducing input. *IEEE Transactions on Software Engineering*, 28(2):183–200, 2002.

[105] G. Misherghi and Z. Su. Hdd: Hierarchical delta debugging. In *ICSE*, pages 142–151, 2006.

[106] J. Clause, A. Orso, and ro. Penumbra: Automatically identifying failure-relevant inputs using dynamic tainting. In *ISSTA*, pages 249–260, 2009.

[107] M. Gulzar, Interl, M. i, S. Yoo, S. Tetali, T. Condie, T. Millstein, and M. Kim. Bigdebug: Debugging primitives for interactive big data processing in spark. In *ICSE*, pages 784–795. IEEE, 2016.

[108] C. Sun, Y. Li, Q. Zhang, T. Gu, and Z. Su. Perses: Syntax-guided program reduction. In *ICSE*, pages 361–371, 2018.

[109] R. Gopinath, A. Kampmann, er, N. Havrikov, E. Soremekun, and A. Zeller. Abstracting failure-inducing inputs. In *ISSTA*, pages 237–248, 2020.

[110] C. Wohlin, P. Runeson, M. Hst, M. Ohlsson, B. Regnell, and A. Wessln. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2012.

[111] SAE On-Road Automated Vehicle Standards Committee et al. Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. *SAE Standard J*, 3016:1–16, 2014.