IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 7, 2020, accepted December 18, 2020, date of publication December 29, 2020, date of current version January 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2020.3048047

Software Verification and Validation of Safe Autonomous Cars: A Systematic Literature Review

NIJAT RAJABLI[1], FRANCESCO FLAMMINI[1,2], (Senior Member, IEEE), ROBERTO NARDONE[3], AND VALERIA VITTORINI[4]

[1]Department of Computer Science and Media Technology, Linnaeus University, 351 95 Växjö, Sweden
[2]Division of Product Realisation, School of Innovation, Design and Engineering, Mälardalen University, 632 20 Eskilstuna, Sweden
[3]Department of Information Engineering, Infrastructure and Sustainable Energy, Mediterranean University of Reggio Calabria, 89124 Reggio Calabria, Italy
[4]Department of Electrical Engineering and Information Technology, University of Napoli Federico II, 80125 Naples, Italy

Corresponding author: Francesco Flammini (francesco.flammini@ieee.org)

This work has been partially supported by the research project RAILS (Roadmaps for A.I. integration in the raiL Sector). RAILS has received funding from the Shift2Rail Joint Undertaking (JU) under grant agreement No 881782. The JU receives support from the European Union's Horizon 2020 research and innovation programme and the Shift2Rail JU members other than the Union.

ABSTRACT Autonomous, or self-driving, cars are emerging as the solution to several problems primarily caused by humans on roads, such as accidents and traffic congestion. However, those benefits come with great challenges in the verification and validation (V&V) for safety assessment. In fact, due to the possibly unpredictable nature of Artificial Intelligence (AI), its use in autonomous cars creates concerns that need to be addressed using appropriate V&V processes that can address trustworthy AI and safe autonomy. In this study, the relevant research literature in recent years has been systematically reviewed and classified in order to investigate the state-of-the-art in the software V&V of autonomous cars. By appropriate criteria, a subset of primary studies has been selected for more in-depth analysis. The first part of the review addresses certification issues against reference standards, challenges in assessing machine learning, as well as general V&V methodologies. The second part investigates more specific approaches, including simulation environments and mutation testing, corner cases and adversarial examples, fault injection, software safety cages, techniques for cyber-physical systems, and formal methods. Relevant approaches and related tools have been discussed and compared in order to highlight open issues and opportunities.

INDEX TERMS Advanced driver assistance systems, automotive engineering, autonomous vehicles, cyber-physical systems, formal verification, intelligent vehicles, machine learning, system testing, system validation, vehicle safety.

I. INTRODUCTION

Accidents on roads happen so frequently they are considered part of everyday life. However, the number of fatal accidents worldwide is not to be neglected. The recent number of casualties on roads exceeds 1 million per year globally [1]. Fatal accidents are considered to be a major problem in many countries. While some countries aim to decrease the number of fatal accidents by 50% in the next 5 years, others, for example, Sweden, have set a goal to completely prevent such accidents [1]. A summary in reference [2] shows that human errors account for 75% of all road accidents, and this number is over 90% in the United States. According to recent studies, these numbers do not show a sign of decline if humans are left in charge to fully control the vehicles. For human drivers, the decision-making is 90% based on visual perception, and as a matter of fact, humans are not capable to be fully aware of potential hazards in the surrounding environment [2]. Additionally, traffic jams are also usually caused by either poor decisions by humans or their inability to accurately monitor the behavior of all other vehicles. Traffic congestion does not only lead to loss of time and resources, but it also causes a rise in air pollution and greenhouse gases. These issues are of paramount importance in industrialized civilizations, and introducing autonomous vehicles (AVs) on roads is seen as a viable solution to these problems. However, the fact that AVs

The associate editor coordinating the review of this manuscript and approving it for publication was Bohui Wang.

provide overall increased safety compared to human driving needs to be demonstrated.

It is rather easy to demonstrate the scale of human mistakes leading to safety issues on roads. However, switching from human drivers to AI systems is currently a challenging transition. AI systems showed impressive results in different domains and are progressing quickly. However, their application to safety-critical domains, such as AVs, creates several V&V issues, mainly due to limited predictability. Although it is possible to statistically show the accuracy of AI systems to some extent, it is most often not possible to fully understand how they came to a specific conclusion. Therefore, these systems should be regarded as unsafe until their safety is proven according to reference standards such as ISO 26262 [3]. While ISO 26262 provides guidelines for the development of safety-critical systems and requirements applicable to the automotive industry [4], it is not yet clear whether this standard can be effectively used as it is in AI-based systems.

Further issues arise when considering more than one AV operating on roads. Even if a single vehicle performs well under test conditions with a reasonably small error rate, deploying millions of AVs may increase the total failure rate to an unacceptable level [3]. However, transition on such a large scale is unlikely to happen at once: the control of human drivers can be gradually reduced in order to progressively reach full automation. The Society of Automobile Engineers (SAE) has defined six levels of automation for AVs. In addition to outlining the responsibilities of the human driver, those levels also help to determine technical requirements to ensure the safety of AVs [5]:

- Level 0. The vehicle system has no control over the vehicle. As the full control and responsibility are on the human operator, the vehicle is only allowed to issue a warning in hazardous situations and cannot command the actuators.
- Level 1. The human driver and the vehicle cooperate throughout the ride. The contribution of the vehicle helps to increase driving performance, however, it is limited to steering or acceleration. The driver should always stay aware of the situation.
- Level 2. The vehicle assists the human operator by fully controlling the vehicle. Although the driver does not need to control the vehicle, he/she should have full situational awareness to intervene whenever needed.
- Level 3. Vehicles classified at this level have restricted decision making abilities and perception through sensors. Although the human driver does not need to have complete situation awareness, he/she is still required to be ready to take control in dangerous situations. The attention of the driver is less strictly required at this level.
- Level 4. At this level, although the driver can intervene, he/she is not expected to take control of the vehicle. Highly Automated Vehicles (HAV) are responsible for all tasks including safety-critical ones and can operate safely without human input in predefined modes.
- Level 5. Vehicles classified at this level can operate themselves in any situation. According to reference [6], at this level, the driver is completely out of the loop, while reference [5] suggests that completely taking control away from the human driver, i.e., removing the steering wheel and pedals, may be optional; in both references, the authors agree that the driver has no responsibility for vehicle control. If the driver cannot intervene, the vehicle should be classified with a high ASIL (Automotive Safety Integrity Level) as defined in ISO 26262 as they lack controllability [3].

Vehicle systems from Level 0 to Level 2 are usually introduced as Advanced Driver Assistance Systems (ADAS). At these levels, the human driver is held responsible for the vehicle's operation and therefore should monitor the vehicle and intervene in unsafe situations [6]. Figure 1 depicts a graphical summary of the described SAE levels.

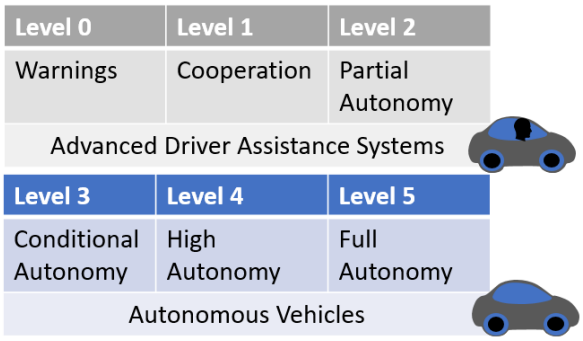

FIGURE 1. SAE levels.

So far, the most advanced autonomous cars introduced to the public are only partially autonomous since a human driver should still stay alert to supervise and take over the control of the vehicle when needed to avoid or manage hazardous situations [4]. A fatal accident in 2018 involving a Tesla car and a truck [7] showed the necessity of human supervision in autonomous cars at the current stage of evolution. The accident happened due to the inability of the car's artificial vision system in autopilot mode to detect a white truck on a cloudy sky background. Such a corner case for the camera was not part of Tesla's test suite [8] and could only be identified by a human driver.

The importance of software V&V for the safety assurance of autonomous cars is the main motivation of the study presented in this paper. The increasing complexity of software systems will raise the cost of production [4] and reliable V&V approaches are crucial as a single bug that goes unnoticed can be extremely costly for a company [9]. Even without AI being involved, currently, around 60% of vehicle recalls are due to software bugs [10].

In such a context, this paper presents a comprehensive Systematic Literature Review (SLR) addressing the state-of-the-art of safety V&V frameworks and approaches for software systems in autonomous cars.

The main objectives of this study are to review existing methodologies and tools used to tackle challenges in the safety assessment of autonomous cars, highlight open issues, and provide some hints about future research directions. This study is also important to define a baseline on which evaluating the potential for technology transfer to other transportation domains, such as railways, where autonomous vehicles can adopt some of the techniques that are being pioneered in the automotive domain due to the huge and growing interest in self-driving cars. This is one of the objectives of the Horizon 2020 research project named RAILS (Roadmaps for AI Integration in the Rail Sector) funded by the Shift2Rail Joint Undertaking, a body of the European Union [11]. Please note that all literature addressing V&V of more common safety-critical automotive software such as the one controlling legacy ABS (Automatic Braking Systems) are out of the scope of this survey.

Since this paper is classified as an SLR, it will only include papers published in reputable sources that are publicly accessible and already indexed in reference literature repositories; for this reason, and due to the extremely competitive automotive industry landscape, it is important to highlight that the study presented in this paper cannot include any research results that are either not peer-reviewed or not made public for confidentiality reasons. Furthermore, since the safety of self-driving cars is considered a ''hot'' research field nowadays, works are continuously added to the reference literature. Therefore, it cannot be in the scope of this paper to provide an always up-to-date picture including all of them, which would be impossible, but rather to define the main research trends by grouping papers into relevant categories and discuss them in terms of main challenges and opportunities.

Please note that in the paper we often use the abbreviation AV for autonomous (road) vehicle, which we mainly considered as a synonym for self-driving or unmanned car, although, as explained above, road vehicles might be of different types and featuring different levels of autonomy; however, it should be clear that we are not considering infrastructure-based ITS (Intelligent Transportation Systems) implementations where connected vehicles (sometimes even framed in the ''Internet of Vehicles'') are controlled centrally and/or through vehicle-to-vehicle (V2V) communications, such as platooning approaches.

The rest of the paper is organized as follows: Section II describes the research methodology and provides a general classification of the papers resulting from the literature search. Section III provides the general results of the SLR serving as background information about the life-cycle and certification of safety-critical software in autonomous cars, as well as the V&V approaches for machine learning systems currently proposed in the literature. Section IV analyses and discusses more specific SLR results, i.e., the state-of-the-art of techniques for the safety assessment of autonomous cars. Section V outlines open issues, challenges, and opportunities as they emerge from the review. Section VI briefly reports about related work in terms of similar SLR, and finally, Section VII closes the paper by summarizing findings and results of the study.

II. RESEARCH METHODOLOGY

This survey adopts the guidelines for Systematic Literature Review (SLR) in software engineering from references [12] and [13]. Following these guidelines, the SLR has been conducted in the two phases ''Plan Review'' and ''Conduct Review'', each one consisting of several steps.

A. PLAN REVIEW

In this phase, the review protocol, the research scope, and research questions are specified. In the following sections, these steps are described in more detail.

1) REVIEW PROTOCOL

The review protocol contains elements of the SLR and is necessary to define the methodology, the review process, and criteria used to filter the selected papers [12], [13]. Components of the review protocol of this SLR consist of research questions, search process, inclusion and exclusion criteria, and classification of the related work. Details of these components start with this section and continue with section II-B.

2) RESEARCH SCOPE

The scope of this SLR encompasses reviewing research work in recent years (approximately, the last ten years) related to the V&V of safe autonomous cars, with a focus on software engineering, to identify the state-of-the-art and open issues. Related SLR are discussed in section VI.

3) RESEARCH QUESTIONS

Research questions of this SLR are defined as follows:

- RQ1: What are the most common requirements in the software V&V of safe autonomous cars?
  The objective of answering this question is to identify V&V requirements according to existing safety standards and regulations. This will also give us an insight into the main objectives of the certification processes of AVs.
- RQ2: What are the main challenges in performing software V&V of safe autonomous cars?
  The goal of answering this question is to present approaches to check that the software in autonomous cars meets given safety requirements. The shortcomings and potential of different methods need to be analyzed and compared where relevant.
- RQ3: What are the open issues and opportunities in software V&V of safe autonomous cars?
  The objective of answering this question is to investigate the state-of-the-art in order to highlight open issues in an attempt to determine promising directions and opportunities for future research.

B. CONDUCT REVIEW

Research work to be reviewed has been found by searching on Elsevier's citation database known as Scopus. The reason to choose Scopus as the reference literature search engine is its wide international coverage of research work and digital libraries from most reputable publishers, plus a quality check preventing the inclusion of low-quality materials. At the same time, Scopus is extremely inclusive: at the time of writing this paper, Scopus claims to have indexed more than 5.000 international publishers including renowned publishers in computer science, such as ACM, IEEE, Wiley, and Springer. For conducting the review, first, an automatic search has been done using a specific search query; after that, appropriate inclusion and exclusion criteria have been defined and applied to select relevant studies from the results.

1) AUTOMATIC SEARCH

In this section, the search query to retrieve relevant studies is defined. The following main query has been used to search in title, abstract and keywords:

safe* *AND* ((self *PRE/0* driving) *OR* (autonom* *PRE/0* (vehicle* *OR* car* *OR* automobile* *OR* driving)))

PRE/0 is a Scopus specific operator to specify that the second word should precede the first one immediately, i.e. there should be no words between two words. The first instance of the operator is used to include both ''self-driving'' and ''self driving'' keywords. In the second instance, the operator is used instead of *AND* operator as *AND* operator would match any text that includes the keywords in any distance between each other. This, for example, would lead to many results with autonomous marine vehicles, autonomous aerial vehicles or autonomous heavy vehicles. In a search engine that does not support the proximity operator, the query would need to be modified to include exact matches such as ''autonomous vehicle'', ''autonomous car'' and their variations. Using the proximity operator *PRE/0* helps us to exclude irrelevant studies effortlessly. Even though the focus of this SLR is on autonomous cars, studies related to other types of vehicles have been included to cover technologies also used in autonomous cars. Adversarial attacks targeting image recognition systems can be an example subject because they are not specific to autonomous cars but also used in other AVs. The general search query allowed us to explore research in different areas that could provide valuable information in support of the V&V of safe autonomous cars.

In addition to the main query, in order to limit search results to software V&V, only studies that have keys matching to the following query have been selected:

(testing *OR* verification *OR* validati*) *AND* (software *OR* ''Artificial Intelligence'' *OR* ''Machine Learning'' *OR* ''Deep Learning'' *OR* ''Neural Network*'')

Although the keyword ''software'' covers most of the related work (75% of the search results), adding common keywords related to software aspects of self-driving cars has increased inclusiveness.

The results are also limited to research work published between 2009 and 2020 in the ''Computer Science'' and ''Engineering'' fields. We got about 200 results for our query. Two of the results have been removed as they were duplicates of references [8] and [14].

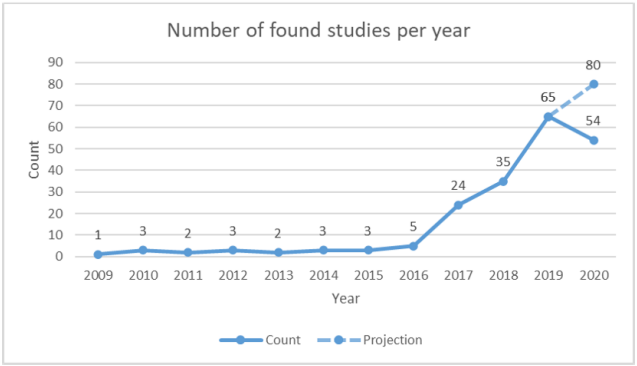

FIGURE 2. Distribution of found studies over the years.

In Figure 2, the distribution of papers over the recent years is illustrated. Starting from 2016, there has been a steep increase in the number of papers published on the reference subjects, which demonstrates the growing interest in the research community. Please consider data for the year 2020 is incomplete as many papers still need to be indexed, therefore the dashed line provides a conservative estimate based on linear regression from 2016, although the growth is actually more likely to become exponential.

2) INCLUDED AND EXCLUDED STUDIES

After determining the initial set of papers to review, inclusion and exclusion criteria have been defined and applied to the results in order to remove false positives and only keep relevant studies. Inclusion and exclusion criteria have been applied mainly considering only title and abstract; however, when the situation was not clear, the paper full text has been reviewed, as suggested by reference [13]. Inclusion and exclusion criteria are defined below.

Inclusion criteria:

- I1: *Studies focusing on V&V of software systems in safe autonomous cars*
  Research work included by this criterion has a focus on the safety of autonomous cars specifically. In contrast with other autonomous vehicles, or autonomous road vehicles to be specific, autonomous cars would interact with other human drivers on busy roads and pedestrians inside the city more frequently and closely. Thus ensuring their safety is different, for instance, than that of autonomous heavy vehicles that usually do not drive in areas with many pedestrians.
- I2: *Studies focusing on V&V of safety-critical software systems in different types of vehicles that is also applicable to autonomous cars*
  This criterion covers studies that focus on safety V&V of software that is used in any type of vehicle and

not just cars. To give an example, references [14]–[17] provide useful information about adversarial attacks that can target any type of autonomous vehicle including autonomous cars, but does not contain the keyword "car" anywhere in the title, abstract or keywords, and therefore would not be normally included.

Exclusion criteria:

- E1: *Studies that do not apply to autonomous cars*
  Studies about AVs that cannot be applied to autonomous cars, such as approaches suitable to aircraft and marine vehicles that cannot be transferred to the automotive domain, are not in the scope of this paper. Also, papers that mention autonomous vehicles, but do not provide further information related to the topic are considered as false positives and are therefore excluded.
- E2: *Studies that do not focus on V&V for safety assessment*
  Many papers about autonomous vehicles mention the importance of their safety, and since those studies are related to software engineering, they also mention verification. However, not all of these studies are about V&V for safety assessment, hence those unrelated are excluded from our study.
- E3: *Studies that do not focus on software aspects of safety V&V*
  Research work that primarily focuses on hardware (e.g., sensor technologies) with no sufficient information provided on software is not included in this SLR.

The process of applying inclusion and exclusion criteria to 200 studies resulted in 105 primary papers. Figure 3 shows the distribution of selected studies over the years: the same considerations hold here as in the case of Figure 2.

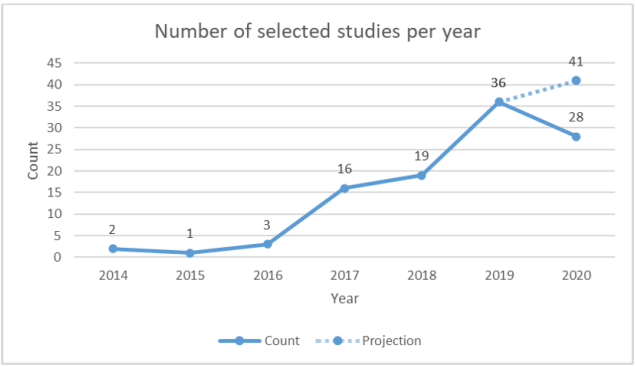

FIGURE 3. Distribution of primary papers over the years.

Figure 4 shows the cumulative number of citations received by the papers in the last decade: again, the graph shows a clearly increased interest in recent years, starting from 2016, with the contribution of a few seminal papers on the subject that we also mention quite often in this study, although citation data for most recent years still needs to be stabilized. Figure 5 shows the geographical distribution of publications based on affiliation of authors. United States lead the research

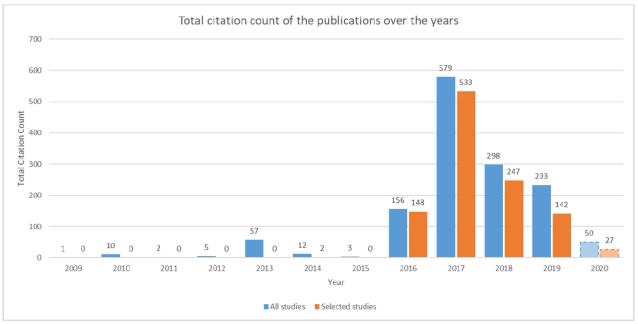

FIGURE 4. Cumulative number of citations of the publications per year.

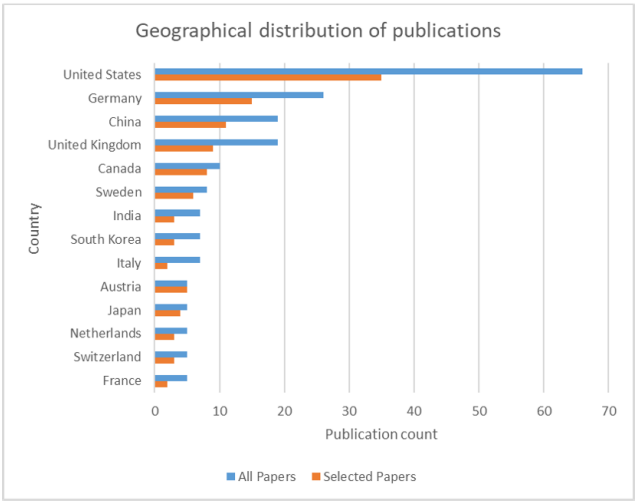

FIGURE 5. Geographical distribution of publications.

on this subject ranking first, not unexpectedly due to the pioneering research on smart-highways and large investments from leading companies like Tesla Motors and Google. Countries with a tradition of high quality academic research and/or strong automotive industry sector, also rank high in the graph.

TABLE 1. Classification according to inclusion and exclusion criteria.

Criterion	Studies
I1	[2], [4], [8], [10], [18]–[55]
I2	[1], [3], [5]–[7], [9], [14]–[17], [56]–[91], [91]–[107]
E1	[108]–[118]
E2	[119]–[170]
E3	[171]–[200]

Classification of studies according to inclusion and exclusion criteria is provided in Table 1. The categorization of the papers according to their main topics is outlined in Table 2, where CPS is the acronym of Cyber-Physical System. In the table, we have grouped papers in homogeneous categories according to the most recurrent topics addressed by the studies, hence following a "bottom-up" approach. We have avoided any classification causing ambiguities, such as the distinction between "verification" (i.e., checking the system against its specification) and "validation" (i.e., checking that

TABLE 2. Categorization of studies according to main topics.

Topic	Studies
Simulation Environments and Test Scenarios	[2], [10], [29], [30], [52], [60], [66], [69], [71], [75], [87], [90], [91], [91], [100], [106]
Test Case Definition and Generation	[17], [25], [36], [40], [46], [51], [52], [54], [67], [72], [81], [86], [87], [100]
Corner Cases and Adversarial Examples	[8], [14], [18], [22], [24], [31], [32], [38], [41], [47]–[49], [53], [55], [85], [89], [95], [96], [102], [104], [105]
Fault Injection	[3], [4], [6], [15], [37], [62], [64] [73] [74], [92]–[94], [97], [103], [107]
Mutation Testing	[4], [50], [55], [58]
Software Safety Cages	[7], [68]
Techniques for CPS	[20], [21], [28], [35], [39], [56], [65], [76], [99]
Formal Methods	[16], [27], [34], [61], [63], [70], [79], [82], [87], [98], [99], [101], [102]

specification is appropriate for the application), because in several studies we found those words have been used with different meanings. It is worth mentioning that there is some overlap in topic coverage of those categories, nevertheless the classification has proven appropriate for the sake of discussion presented in Section IV, where those categories will be described in more details. We attempted to use different categories with a ''top down'' approach, based, e.g., on life-cycle stages, architectural levels or components, but those classifications did not match well with the coverage of papers we actually found, due to several factors, including heterogeneity in reference architectural models and immaturity of new V&V approaches for AVs, compared to more stable sectors such as avionics or railways (e.g., automatic train control), where those classifications work better.

III. GENERAL SLR RESULTS: OVERVIEW OF SAFETY ASSESSMENT IN AUTONOMOUS CARS

In this section we provide an overview of safety assessment in autonomous cars, serving as background knowledge for more specific techniques, by exploring certification challenges as well as the most general and seminal studies on the subject, as emerged from the SLR.

AVs brings new challenges in safety assessment. There is a plethora of approaches and techniques for the V&V of conventional software systems; however, autonomous systems diverge from ordinary systems as they learn, adapt, and change according to new situations [7]. Therefore, traditional methods cannot be directly applied to autonomous systems. Reference [42] argues that the lack of sound methodology to assess the safety of such systems may hinder the delivery of AVs to the public. In the following sections, we provide an overview of safety assessment in autonomous cars, as emerged from the SLR, including background information on certification of AVs, challenges in the assessment of machine learning (ML) technologies used in those systems, and approaches proposed for tackling those challenges.

A. CERTIFICATION

ISO 26262 is a widely used standard for safety-critical road vehicles including at least one electronic component.

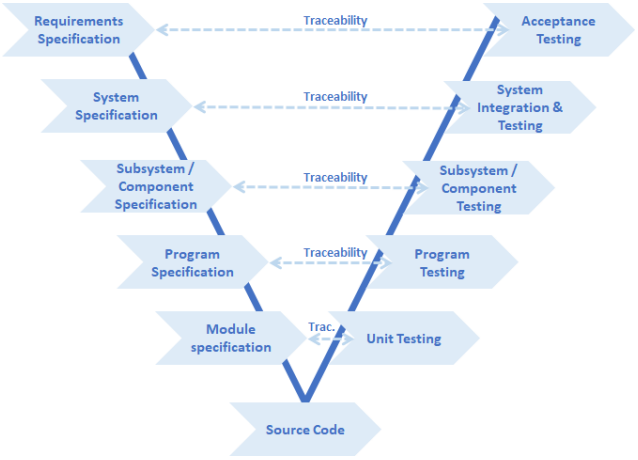

FIGURE 6. V-model in ISO 26262.

The standard conforms to the V-model for product life-cycle V&V management, and provides guidelines for the design and integration of both software and hardware components [4]. The standard provides specifications for model-based development and usage of fault injection for software and hardware components [4]. The V-model used for software development and testing has been applied to vehicles for more than 20 years [3]. The process is shown in Figure 6. Within the V-model, the right side describes the incremental V&V process applied to the waterfall development model described in the left side. This approach relies on the assumption that system requirements and specifications can be defined correctly and exhaustively at well-defined stages. However, such a model is often considered too much ideal because it deviates from actual industrial practice, and that is even worse with AVs due to their extremely dynamic engineering processes [3]. In fact, although ISO 26262 provides general guidelines and best practices for the safety assessment of road vehicles according to the V-model, AVs pose new challenges as mapping the actual development process of AVs to the V-model creates several technical difficulties.

Scaling the classical V-model approach for AVs is especially challenging and does not seem feasible in the near future. The main reason is that in the classical approach a list of requirements should be provided; however, due to their unpredictable nature, AVs may encounter an incalculable amount of possible scenarios. Therefore, all requirements cannot simply be put in plain documentation. Although it might be tolerable that AVs will not be required to handle all rare cases, the requirements should clearly show what is covered and what is not [201].

According to reference [4], the main issue in checking that a system complies with ISO 26262, is to statistically demonstrate that the system will remain in agreement with the safety goals while operating. This becomes even more challenging when considering a fleet of autonomous cars on roads. Even if testing methods show a low statistical failure rate for a single vehicle, when considering, for instance,

a million vehicles on roads, the rate might reach an unacceptable level. To statistically prove that a million vehicles can be on roads one hour each day without having a fatal accident for a thousand days, a single test should run for at least 10^9 hours [3]. ISO 26262 standard has determined the limit for the acceptable ratio of faults as 10 FIT (Failures in Time), meaning if the system operates for 10^9 hours then the number of faults observed should not exceed 10 [6]. Accordingly, to statistically show that the requirement is met, more than one test should be run. However, the fact that such extensive test suits cannot be physically conducted in public traffic for safety reasons poses serious obstacles to the automotive industry.

B. CHALLENGES IN ASSESSING MACHINE LEARNING

The usage of ML systems in industry is mainly based on the so-called Inductive Learning that relies on training data to create models. Identifying pedestrians using camera images can be an example of one of the many use cases of Inductive Learning in AVs. An extensive amount of images are fed to a classifier algorithm which ''learns'' to detect pedestrians and provide a probability of correctness. For assessment purposes, the main objective is that the chosen images are within the system requirements. To test how the resulting model performs, some data from the training set can be kept to be later used to check the system. The process can be mapped to the V-model if training data is associated with the left side of the model at requirements specification level; consequently, on the corresponding right side of the model, requirements can be validated by using the subset of initial training data that was kept out to validate the system. There should be no coincidental correlation between the training set and the expected result to avoid over-fitting. Likewise, the selected validation set should be diverse and should not correlate with the training set while conforming to the requirements in order to detect over-fitting. There is no common way to prove that the resulting ML model for the system is not over-fitted as a safety requirement [202].

One of the limitations of the validation of ML systems is the cost of labelling the data. Labelling is either done by someone or some other unsupervised learning algorithm, both having their own complications. ML systems are sensitive to change: doing a minor change leads to a necessity to re-validate the whole system. When a problem in the training set is identified, it also adds the extra work of collecting more data to validate the system. AVs are likely to encounter many extremely rare cases that were not considered in the initial training set. Every time a new case is detected, the system should be updated and re-validated accordingly [3]. In the next sections, generating test data synthetically within a simulation and the issues that come with such approaches are examined.

Another challenge on the validation of ML is its incomprehensible internal structure for humans. For instance, it might not be clear why a Convolutional Neural Network (CNN) algorithm provided a certain decision or what it has learned about the rules while making a decision. The so-called ''explainable AI'' (XAI) methodologies have been defined to cope with those issues, but in general, making ML easy to understand for humans has not currently been achieved. Being unable to predict the behaviour of ML, makes it harder to assess its decision-making process and leaves us to use costly brute force techniques. Even if a brute force can be applied given sufficient resources, it only validates the result within the training set and does not show the coverage or accuracy of the training data and how it complies with the safety standards. Referring to the example of pedestrian detection, it might be the case that an insufficient amount of sample images where pedestrians in wheelchairs are used, and as a result, the algorithm will not classify those cases as pedestrians [201].

Reference [3] also suggests an alternative way to validate systems that rely on Inductive Learning. The monitor-actuator approach mentioned before can also be used in the validation of such systems. The objective is to make the high-ASIL monitor a deductive component, while the low-ASIL actuator remains an inductive component. Here, the main validation process would be shifted to a component that does not rely on Inductive Learning, thus making the solution of the validation problem easier. As before, the monitor can observe and catch faults in the functionality of the actuator within a safety envelope to achieve fail-safe operation. Consequently, the safety issue is changed to an availability issue, as the system becomes unavailable but remains safe.

Reference [77] studies a use case of an object detection system that uses ML algorithms and provides an efficient approach to validate the system. For an image recognition algorithm to function properly in safety-critical automotive applications, a failure rate as small as 10^{-12} should be achieved. Validating a system that needs to have such a small failure rate is costly and may take several months of testing in real life. Emulators can be used to generate synthetic data, rather than collecting real-life data through test drives, in order to speed up testing. The problem is that the physical implementation of the system does not yet exist to rely on when validating a system at the design level. In this case, simulation approaches should be used, which in turn generates high costs if the model of the system is required to be accurate. To reduce validation cost in such a system, reference [77] proposes an approach of subset sampling in which the objective is to estimate the failure rate of an ML algorithm in an AV in terms of different probabilities. By experimenting within the use case of STOP sign detection, the study concludes that the proposed approach is 15.2 times faster than the traditional brute-force Monte Carlo approach without leading to lower accuracy in the result.

C. VERIFICATION AND VALIDATION METHODOLOGIES

The process of ensuring the safety of AVs is considered to be an interdisciplinary challenge [33], [84]. It is crucial to have a sound engineering process to be followed by contributors with diverse skills and areas of expertise. The V-model is

widely used in the automotive industry for the development of safety-critical components [45]. In the V-model, phases of the development process create a V-shaped diagram as illustrated in Figure 6: on the left, the design process follows a top-down approach, while on the right assessment is done in a bottom-up fashion. In practice, the process can be iterative and may not necessarily strictly follow the sequence. Despite having a straightforward sequence of phases, the development of ADASs and AVs can encounter certain issues in different development phases due to their unpredictable nature related to AI [45].

As already mentioned, reference [3] suggests that one of the strategies to assess risk according to ISO 26262 is to implement a monitor and actuator architecture. In this architecture, the actuator module executes the main functionality and the monitor module carries out the validation process. If there is an issue in the actuator, the monitor is expected to stop the whole function including itself, which leads to a fail-safe system. If the pairing of the two modules is done correctly, as long as the monitor module provides high enough ASIL and can identify the faults within itself, the actuator can be modelled in a low ASIL. The monitor is required to catch faults within its own module in order to avoid the scenario where the fault in the monitor prevents the monitor from detecting faults in the actuator. The main objective here would be to simplify the monitor module as much as possible as it is expected to have high ASIL and consequently requires stricter assessment. Therefore, the complex features are implemented in the actuator which has low ASIL. The fail-safe operation of this pair is a great benefit, however, it has its weaknesses. Pairing two different modules, in which a monitor observes an actuator, is aimed to prevent the actuator from executing unsafe instructions. The problem with this approach is that if something wrong happens during operation, the actuator will become unavailable, which is not always intended (e.g., shutting down steering while the vehicle is still on the move). Additionally, implementing this architecture requires more than one pair of monitors and actuators to cross-validate. The design should also be diverse to avoid software related failures to lead to the failure of the whole system. Reference [3] gives Arianne 5 Flight 501 as an example for such a case, in which the same exception led to the failure of both the main and the backup system, as both were unable to handle the exception in a similar way. As much as it is important to have diversity in the components, it is not easy to achieve it. The reason is that if the same requirements are used, it is likely that components designed according to the same requirements will have the same defects. Reference [3] argues that an approach that will guarantee fail-safe operation by detecting faults in the system is necessary to be adopted in safe autonomous systems, and proposes extending the monitor and actuator architecture to be the main strategy in AVs for this purpose.

Reference [45] proposes a new approach utilizing ML and Deep Neural Networks (DNN) for assessing ADASs and AVs both in lab environments and in the real world. According to reference [45], the primary challenge of using the V-model is the lack of efficient information exchange between different phases and between real-world and lab tests. The methodology proposed in that work aims to categorize and group the real-world test data according to functionalities using ML and DNN algorithms. Data can be reused to recreate real-world test cases inside a simulation. Data is also generalized and stored in a common database to be accessible in all phases of the V-model and therefore to be suitable for different original equipment manufacturers (OEM). The paper also illustrates the use of AI-core (ML system) for test case and scenario generation for both laboratory and real-world environments. Reference [45] suggests that AI-core can be used to validate high levels of safe autonomy with minimal human interference.

A novel approach from reference [42] suggests an iterative process in order to solve the AV validation challenges and achieve SAE Level 5 autonomy. The proposed methodology aims to limit AV decisions to those which are validated and can be guaranteed to be safe, i.e. the vehicle can execute actions only inside a validated safe space. The paper demonstrates a control architecture that can be applied to vehicles in any level of autonomy starting from Level 1. Reference [42] suggests that an AV with Level 2, which requires full situation awareness from the human driver, can initially be released and using the proposed approach assess itself while operating. The deployment starts with Level 2 as it initially does not cover the validation space completely. However, by time, as the vehicle learns, it can increase the coverage of the validation space by assessing itself in an iterative manner with, the goal of achieving Level 5, i.e., full autonomy.

Using feedback from many subsystems to cross-check the output is already used in hardware components and sensors, such as radar and vision systems. This reduces the chance of failures if the components are implemented independently. The same approach can be used to validate the decision of AI algorithms. However, assessing such a combination of systems might need billions of tests as the failure rates will be extremely low. Each component can be isolated and validated separately, however, and the system should also be assessed to show that failures happen independently among the components [3].

IV. SPECIFIC SLR RESULTS: TECHNIQUES FOR THE SAFETY ASSESSMENT OF AUTONOMOUS CARS

In this section we address more specific techniques for the safety assessment of autonomous cars, following the main categories and related groups of papers as reported in Table 2, which will be discussed in dedicated subsections.

With the emergence of AVs, there is an increasing necessity for more effective approaches in verifying software technologies and algorithms. There are two main categories to consider to ensure safety by checking if the software behaves as intended. One of them is simulation-based and commonly known as testing, where the real system or its model is run in a virtual environment and the output of simulation runs is

used to demonstrate that the system functions as expected. However, testing cannot prove completeness: if testing could not find an issue in the system, it only means that the system operated correctly for the test cases considered, but does not conclude that an issue does not exist in the system if other scenarios are considered. Another approach to verification based on formal methods can be used to achieve both completeness and soundness. Formal verification relies on mathematical models to prove or disprove specific specifications and properties. What separates it from testing is that the formal verification method is able to find an erroneous state in the system if it exists, and if it exists it can be demonstrated that this is an issue in the system model [18].

Formal verification methods can especially be valuable to verify safety-critical Machine Learning (ML) systems which are usually impractical to test with conventional testing approaches. There is ongoing research on the topic of using formal methods for such systems. The studies that focus on this topic can be divided into two sections by verification of component and system levels. In formal verification, the state space can increase immensely by increasing the number of parameters and possible values they may take. This makes verification of systems relying on Neural Networks (NN) especially difficult as such architectures can have millions of parameters. Satisfiability Modulo Theories (SMT) and Integer Linear Programming (ILP) solvers offer sound solutions to the verification problem of such systems. Regarding SMT, current research consists of applying formal verification on input and output specification of nonlinear NN architecture in the form of a piecewise linear function called Rectifier Linear Unit (ReLU). Given inputs, this approach can provide guarantees for a set of outputs from the Deep NN (DNN) algorithm. However, it is challenging to apply the approach to NN as one needs to take into consideration all stages of more than a thousand ReLUs that appear in such architectures. The ReLU function can be relaxed to reduce the number of phases by excluding ones that are not breaking the specifications. Using semi-formal verification together with testing is also an amenable approach in dealing with ML systems. The goal of these methods is to find corner cases in ML algorithms which may lead the system to an unsafe state. There are also proposals of various scalable algorithms such as random sampling, generative adversarial networks, and node coverage that can generate different inputs to find such cases [18].

A. SIMULATION ENVIRONMENTS AND TEST SCENARIOS

In order to ensure the safety of autonomous cars, they need to be tested in different conditions and environments. Doing these tests in real life has several complications such as public safety, cost and time. Additionally, it is highly unlikely to get exactly the same scenario every time, hence reproducibility is also an issue. Approaches based on simulation environments are a viable solution to test autonomous cars in real-world alike environments where any scenario can be synthetically generated and repeated [46].

Reference [46] focuses on how different weather conditions, in specific rain that can obstruct the camera's perception, affect the accuracy of decisions in AVs. The proposed approach in the study utilizes the Continuous Nearest Neighbor search together with an R-tree data structure. Realistic 3D vision with raindrops is created using stereo images for the simulation. The approach can be used on already existing images so that the training and test images for the algorithms do not to be recaptured for rainy weather and can be modified to reflect such conditions. For the experiments in the study, Recurrent Rolling Convolution (RRC) object detection network was used with the KITTI data set at Karlsruhe Institute of Technology. Testing on 450 images, the results show that when introducing raindrops to the scene, measured average precision of the algorithm drops by 1.18% and the overall average accuracy drops by 0.37%. Reference [46] also demonstrates that detection accuracy for smaller objects is more sensitive to obstruction by the raindrops (measured average precision for pedestrians drops by 4%).

According to reference [66], one of the challenges in fully utilizing simulation in development is the lack of a test criterion to abstract the real-world environment and automatically generate test scenarios reflecting road conditions. Reference [66] investigates research directions to tackle these challenges. The study focuses on defining test criteria on environmental factors, roadway designs, and dynamic object behaviors, and then how test scenarios can be generated according to these criteria.

A prototype called AsFault to automatically create synthetic test scenarios in a systematic way in simulation is proposed in references [29], [30]. The tool uses a search-based procedural content generation technique and is demonstrated by tests performed on lane-keeping functionality in an autonomous car program. The objective of the tool, in this case, is to create conditions that lead the car out of the roadway. The experiments done with the tool show that it is able to efficiently generate cases where the autonomous car may violate the requirements. The study explains the process of virtual road generation in detail and concludes that more future work is to be done to generate more physically realistic roads by using real-world roads and terrains as an input. The authors also note that currently one lane-keeping system is evaluated due to the lack of availability of such systems and more tests with various execution conditions need to be performed to further confirm the validity of the tool.

Reference [75] proposes a framework called MOBATSim that integrates fault injection and simulation together. The framework is developed in MATLAB Simulink and provides ways to model different real-world scenarios in simulation. The advantage of the approach over other simulation-based approaches is that it supports injecting faults at run-time to verify whether safety requirements are violated in the simulation. The framework is tested on two sensors, front sensor (used for platooning) and speed sensor as a case study. First, a simulation without injecting any faults is run to create a reference point. Then for the front sensor, measurement noise,

and for the speed sensor value, stuck-at faults are introduced. In the case study, 400 simulations are run with faults, increasing the level of faults for each sensor to measure the vehicle's tolerance for faults and a detailed comparison of the results is presented. The authors aim to extend the framework to make it a full safety evaluation tool in accordance with the ISO 26262 standard.

Reference [40] presents a Domain-Specific Language (DSL) called GeoScenario to reduce the cost of creating test scenarios for simulation. According to the study, the main motivation behind designing a DSL for this purpose is that the current simulation tools require engineers to learn different tools and then program everything by themselves. Reference [40] argues that having a DSL that is not dependent on any tool makes it much easier to transform existing scenarios to formats understood by different simulation tools. The language relies on the Open Street Map standard and can be extended based on requirements. GeoScenario is then used in an autonomous car project to show its usage in practice and the authors hope that using a common language will increase the usability of test scenarios designed by different researchers. A similar work carried out in reference [51] focuses on the use of ontologies to achieve the same goal of simplifying and standardizing test case generation for simulation of autonomous cars.

B. TEST CASE DEFINITION AND GENERATION

One of the major challenges in the testing of AVs is that the system behavior is strongly dependant on its environment: as all possible cases that might happen in the real world cannot be predicted, the system will constantly learn and adapt during its lifetime. Therefore, it essential to have a run-time testing approach to test the system continuously [25]. In reference [25], theory and application of a framework to achieve real-time testing in autonomous cars is described. The study especially focuses on the Internet of Things (IoT) and mentions the importance of real-time testing and its challenges within IoT and autonomous systems. During operation, autonomous cars may have a very small amount of time to do testing when experiencing a new situation. The authors demonstrate how Combinatory Logic can be used to generate new test scenarios for the extended system from test scenarios designed for the initial system.

Another approach dealing with performance and variety is proposed in reference [81]. Initially, training inputs representing different traffic situations are defined. Then the objective is to define test cases by considering the actions of other objects in traffic. Abstract definitions are used to classify these actions and scenarios. This way, test cases can be compared to eliminate identical cases from the test. However, the approach requires manual efforts when identifying relevant scenarios, thus it needs to be fully automated using ML to make it practical. The authors note that the effectiveness of the approach needs to be evaluated after its automation.

As previously mentioned, the number of possible scenarios in the real world is virtually unlimited, thus it is not possible to test every possible case. To cope with this problem and reduce the testing space, reference [17] proposes an approach to determine circumstances that might lead the system outside its safety limits. Reference [17] argues that although Hazard Based Testing (HBT) approaches are able to discover test cases that make the system fail, they cannot show the specific parameters that cause the failure. In the proposed approach, combinations of parameters in the test cases that cause the failure of the system are identified with Bayesian Optimization. It then is implemented in a simulation model to demonstrate how finding multiple combinations of parameters help to discover testing conditions easier than using random testing methods.

Simulation, especially when considering 3D simulation with realistic details, requires extensive computation power and therefore is costly. In order to reduce testing space and the number of required tests to be run consequently, search-based testing algorithms can be implemented [67]. As in previous approaches, the objective is to find only those scenarios that lead so safety violations. Reference [67] compares two search algorithms, namely the genetic algorithm and simulated annealing to evaluate their effectiveness in the domain of testing autonomous cars. Both algorithms are run in a simulation framework and time-to-collision is measured to be used as a termination condition. The results of the study show that using the genetic algorithm, safety-critical cases can be generated by performing a lower number of tests than when using simulated annealing. Furthermore, previously in reference [36], it has been shown that the genetic algorithm is also more effective than random selection in generating such cases.

Reference [19] proposes an approach to generate test cases for features that are independent of each other in terms of functionality yet interact with one another. In order to identify feature interactions that lead to violation of safety requirements, a search-based algorithm, named FITEST, is developed. A case study and experiment settings are provided together with the results. The results from two systems and the discussion with the system developers show that failures found using FITEST were not identified by the testers before and were indeed related to feature interactions [19].

Reference [23] proposes an approach that uses DNN's sentiments as a way to prioritize inputs that can lead to erroneous behavior. Three sentiments are considered in the study: confidence, uncertainty, and surprise. These measures are evaluated to demonstrate their efficiency in identifying cases that can lead to failures. The results from the experiments on MNIST models show that the approach can identify safety-critical cases with a varying 88% to 94.8% accuracy and that the sentiments can indeed be used as a way to prioritize inputs. Reference [23] notes that the method is not only suitable to reduce testing cost during development but can also be implemented to predict safety-critical cases and activate corresponding mechanisms for protection in real-time.

C. CORNER CASES AND ADVERSARIAL EXAMPLES

Deep Learning (DL) systems refer to systems that comprise one or more DNN elements or consist of entirely DNNs. Lately, the accuracy of achieving tasks that utilize DL, such as object detection and speech recognition has improved significantly, to a level that it can even outperform humans in some cases. This progress has contributed to the use of DL in safety-critical components of autonomous cars on a lager scale [8].

However, despite those improvements, behavior and decisions of DL algorithms are sometimes unpredictable, or even wrong. Such corner cases might exist due to issues in the model or training data, for instance, overfitting or underfitting of system model, or bias, or lack of diversity in the training data [8]. Additionally, DL systems can be vulnerable to adversarial attacks that make minimal changes in the input in order to cause, for instance, an image classification algorithm to wrongly classify the traffic sign, leading to a wrong decision that might result in an accident. Such perturbations on images may be invisible for humans and need to be discovered during testing [38]. Sometimes such corner cases happen without someone intentionally changing the input, but due to weather conditions or differences in lighting degrading the perception of the camera [32]. Therefore, while working on a safety-critical domain, it is essential to have a sound methodology to test for corner cases in order to avoid accidents that may result in great damage [8].

As noted in references [8], [38], currently the common ways of testing DL components comprise collecting a vast amount of real-world data and labeling it manually to create a test set, or generating data synthetically using simulation. In addition to being expensive [38], these common practices do not take into account the inner mechanisms of DL algorithms. Considering the number of possible scenarios in the real-world, such testing methods might not cover the whole input space and will certainly miss the majority of corner cases. Therefore, even if all tests pass successfully, the result cannot be used to prove that there is no issue in the system, as previously mentioned in reference [18].

To cope with the aforementioned issues, reference [8] proposed design, implementation, and evaluation of the first white-box testing framework for the industry, named DeepXplore. The framework aims to achieve systematic testing of real-world DL applications. The presented approach presents neuron coverage as a metric that is used to measure the proportion of the active neurons in DNN to increase the effectiveness of tests. The neuron coverage concept can be compared to conventional code coverage in a way that in code coverage parts of the code that are invoked by the input is measured, while in neuron coverage activated neurons are measured. Although the implementation of DL itself is a software, the model that is doing the main task is not developed by a programmer but is a result of the training data. Reference [8] notes that it is rather easy to reach 100% code coverage while the neuron coverage does not exceed 10%. Later in the paper, several DL components with matching features are used together to cross-check the decisions, which eliminates the need for checking the result manually. Reference [8] also demonstrates a methodology to find corner case inputs that might activate incorrect behaviors, and increase neuron coverage by using gradient-based search approaches. According to reference [8], DeepXplore can effectively discover corner case inputs in the most advanced and complex DL systems (fifteen of them are tested in the paper). The average performance of DeepXplore on a regular notebook shows that it is capable of finding one corner case per each DL model in a second. It is also shown that the generated corner cases can be reused as a training set to train the DL system again to increase its accuracy about 3% [8].

Despite the promising results of the DeepXplore framework, there are 2 major threats to its validity. One is pointed out in reference [38] that cross-referencing multiple models is not a reliable approach as currently, attackers are able to generalize the adversarial example for these models to trick the system. According to experiments in recent studies, another criticism towards the framework and usage of neuron coverage in general has emerged. Reference [58] notes that even achieving 100% neuron coverage is not enough to verify the safety of DNN applications as maximum neuron coverage is proved to be reached by merely using specific input vectors from the training data.

Following DeepXplore [8], another testing framework called DLFuzz, which also utilizes the differential testing technique, is proposed in reference [31] to find corner cases in DL systems. DLFuzz iteratively mutates the input to increase neuron coverage and tries to maximize the difference between results (predictions) per input in order to find rare inputs. After finding inputs that lead to an increase in the neuron coverage, the process continues to find an invisible perturbation by mutating these inputs. The advantage of DLFuzz over DeepXplore is that DLFuzz does not require to label test set manually or cross-check the results with other DL systems. As a result, DLFuzz outperforms DeepXplore's white-box testing by finding 338.59% more adversarial examples that have 89.82% smaller perturbations. DLFuzz also achieves to increase neuron coverage 2.86% more in 20.11% less amount of time. However, reference [31] has observed that in tests for the ResNet50 neural network, which has a much higher number of neurons, DeepXplore was faster as DLFuzz had to spend more time selecting neurons. Nevertheless, the adversarial examples found by DLFuzz contain perturbations too small to notice with a human eye, while the ones generated by DeepXplore can be clearly visible.

Another differential testing framework for DNN systems, called DeepHunter, is proposed in reference [14]. Similar to DLFuzz it aims to preserve the semantics of the input while mutating it into an adversarial example. Additionally, the paper proposes a seed selection approach that couples variety and recentness. The variety of seed selection is achieved by decreasing the probability of tests that have been

fuzzed before, and recent tests are biased by giving them more priority to balance recentness property. Reference [14] demonstrates that while finding defects in the DNN system, DeepHunter is capable of preserving the semantics of the initial tests with a validity rate of 98%. The authors also conclude that the seed selection with a more priority to diversity has more influence than recentness-based selection on increasing neuron coverage. It is also illustrated that DeepHunter surpasses DeepTest and TensorFuzz approaches in coverage, and the amount and variety of adversarial examples found. A recent work in reference [55] presents a new framework called DeepSmartFuzzer and argues that it outperforms both DeepHunter and TensorFuzz. The experiments in the study use DeepXplore's neuron coverage as well as four other coverage criteria to compare the frameworks. Source code is also provided to enable researchers to reproduce the experiment results.

Reference [38] proposes a different approach to solve the oracle problem (humans needing to manually label inputs and check results limiting the extensibility of testing) without relying on cross-referencing the results. The approach utilizes a metamorphic testing methodology in which the accuracy of the test is measured by the change in the output. This relation between the input and the output is referred to as Metamorphic Relation. Using the metamorphic relations approach to solve the oracle problem also eliminates the necessity of using ML algorithms, which are expensive to execute, for finding adversarial images. Reference [38] argues that using metamorphic relations is the most efficient way of finding adversarial inputs, and can help to achieve significant results with an accuracy of about 90% on average and 96.85% on best-case in image classification.

In reference [32], an approach that relies on Satisfiability Modulo Theory to verify neural network systems is presented. Although the study concentrates on image classification, reference [32] argues that it can be used in different applications that utilize neural networks. The proposed approach aims to ensure the safety of the classification algorithm by finding manipulations that would make the image fall in the same category for the human observer while it would result in a different decision by the algorithm. Examples of such perturbations can be rain, fog, different light angles or conditions. The results of the framework are promising as it can find adversarial inputs in a few seconds. However, the complexity of the process of verifying the algorithm increases exponentially as the number of features increases. According to reference [32] performance issues of the framework can be solved using parallel computation.

Many algorithms that are used to verify DNN require knowledge about the inner workings of the system. Reference [47] proposes a method using a black-box approach to verify image classification algorithms. Key features are taken from the images using an object detection algorithm and the pixels in those parts are prioritized according to their contribution to creating the whole visual meaning of the image for a human eye. The process is implemented using a two-player gaming approach in which each player takes a turn and aims to achieve a specific goal. Here the first player tries to decrease the distance between the original input and an adversarial one by manipulating the image while the other player will either help the first one achieve the objective or will oppose it. Reference [47] argues that in theory, a player-based approach in this fashion can eventually find the smallest perturbation possible to achieve an adversarial image. The authors also show the cases for Lipschitz networks in which safety guarantees can be made about the absence of adversarial images. According to the paper, utilizing a Monte Carlo search, the black-box approach can perform as good as the most advanced and complex white-box approaches.

Reference [48] extends the work in reference [47] by developing a tool called DeepGame that explores two problems concerning pointwise robustness, namely maximum safety radius and feature robustness. In the first problem, the objective is to calculate the minimal distance (in terms of pixels for an image) between the initial input and an adversarial one, hence to find a safety distance from the original input in which no adversarial examples will be present. In the second problem, the goal is to find a safety radius, in which it is possible to control the occurrence of adversarial examples by limiting perturbations to selected features. The computation of these problems is done using a two-player gaming approach, in which the first player picks features and the second player applies manipulations to the input in order to decrease the distance to an adversarial example and the process continues to change the input in an iterative fashion. As in reference [47], the second player can be both cooperating with the first player or opposing it. If players are cooperating the prize of the first player is the maximum absolute safety distance and when the players oppose each other, the prize of the first player is measured by feature robustness.

As mentioned previously, the adversarial examples do not only include intentional manipulations but also situations that can frequently occur in real life such as extreme weather conditions and infrastructure issues on roads. While they might not hugely affect human diver's perception, they can degrade the performance of DNN image classification systems significantly. Reference [49] proposes a framework called DeepRoad that uses unsupervised learning techniques to synthetically generate real-world scenarios for the camera. DeeRoad is able to add different weather filters to the image such as rain, fog and snow, and also can create fake holes on the roads. As in reference [38], DeepRoad also utilizes the metamorphic testing approach. The experiments of the framework are run with the Udacity data set and show its ability to find thousands of inconsistencies. However, the authors note that rather small size of this data set can be one of the threats to the validity of the framework's effectiveness. Additionally, the capabilities of the used Generative Adversarial Network for producing synthetic images should also be kept in mind as it sometimes does not preserve the semantics of some objects in the scene [49].

DeepRoad is later used as a baseline approach for input validation in reference [89]. The authors of this study present a new framework called SelfOracle. Their experiments show that SelfOracle is more than two times effective at discovering misbehaviors compared to DeepRoad both in terms of false positive and true positive rates. They also argue that SelfOracle has a significantly less computational cost in contrast to DeepRoad and provide the source code of the experiments for reproducibility.

Another approach that enables automatic detection of erroneous behaviors in DNNs without manual labeling efforts is proposed in reference [41]. The study focuses on the object detection system and uses the difference between the outputs of two inputs that fall in the same category as a hypothesis to identify false negatives. Two specific cues are used in the proposed system: temporal and stereo. Object detection algorithms that use CNN sometimes fail to identify objects, however, region-based trackers are able to trace the objects within continuous frames. Using previous frames, missing objects in the next frames can be detected, which enables engineers to find the temporal inconsistencies in the algorithm. Stereo inconsistencies refer to the different outputs from the algorithm for the objects in two similar images. Using mapping between images, the location of missing objects can be found. Since the approach can be applied to unlabeled data and object detectors that are required to implement the framework are already a part of autonomous cars, the framework can add an extra layer of verification for finding false negatives with a minimal cost. Although the proposed framework was able to identify the mistakes of the most advanced object detectors in the experiments, the authors note that if no object is detected in the scene or the scene is too crowded for the object detector, the system will not be able to identify the errors. Thus, the framework is currently limited to the capabilities of the object detection algorithm it uses.

D. FAULT INJECTION

AVs include a combination of software and hardware components that makes it more challenging to verify their safety. Since events that can lead to a system failure may happen randomly, it is not only hard to predict them but also to (re)produce these scenarios [15]. To meet the 10 FIT requirement of ISO 26262, not only system permanent faults, but also transient faults, due to e.g. electromagnetic interference and cosmic radiation, should be detected [6]. There are many in-depth approaches such as the one in reference [8] to detect permanent faults, while approaches dealing with transient faults are rather immature in the AV domain, and need to be handled in real-time [6]. In order to test and possibly increase the coverage of fault-tolerance, Fault Injection (FI) can be used. FI is used to insert faults in places where fault/error handling happens [15]. Within safety assessment, FI can be applied at several abstraction levels, even to validate safety arguments. Irregular inputs are given to the system to find its weaknesses, i.e., cases where the system has unexpected behavior. Fault injection can be applied at each different phase of the V-model in ISO 26262.

Reference [73] proposes a methodology utilizing FI at the software level according to ISO 26262. The main objective of the approach is to validate system performance not using synthetically generated data, but with real-world samples. The study concentrates on inserting time-based, sensor data-based and signal data-based faults. The approach is then applied to an object detection algorithm utilizing a camera sensor to see how system reliability changes when faulty inputs are introduced. In the specific use case, ''salt and pepper'' noise is added to evaluate its effects on system performance.

The framework called DriveFI is able to modify the state of software and hardware components to demonstrate the effects of faults in a simulation environment, and use ML-based Bayesian FI to find faults where there is a high chance of violation of safety requirements [15]. The study uses three models to demonstrate the efficiency of the proposed approach. Reference [15] shows that using random FI over 98,400 faults would take 615 days to complete, while with the proposed method it took only 4 hours. Additionally, running in real-time, DriveFI could detect 561 faults with a safety risk, while the random method could not find any in several weeks.

Another FI framework that relies on Systems-Theoretic Process Analysis is introduced in reference [74]. The study introduces the framework using the Openpilot agent in various weather conditions where faults happen at sensors of the system. Hazard analysis is done to produce test cases that might violate safety requirements in order to improve the coverage of testing. The output from the experiments shows that the proposed approach (with 35.78% hazard coverage) is more effective than random fault injection (with 28.97% hazard coverage). The authors provide detailed experiment data, however they suggest more experimentation is needed before making a conclusion about the results.

An approach to measuring fault resilience of ML systems, TensorFI framework, using TensorFlow, is proposed in reference [37]. With flexibility and portability in mind, the framework helps to discover the correlation between various parameters or algorithms, and error resilience of ML systems. Within the experiments in the study, it is demonstrated that the resilience of the system may vary significantly depending on the used algorithms and the inputs. Reference [37] introduces a metric called prediction accuracy drop that shows the difference between original prediction accuracy of the trained model and the accuracy when TensorFI is used; it also demonstrates the relation between accuracy drop and the number of classes in the output data. The authors suggest that exploring such correlations can help with the design of ML systems requiring higher resilience.

Reference [6] extends the work in reference [37] and creates an effective framework called BinFI to find safety-critical bits in ML systems by using TensorFlow. Reference [6] argues that most common ML algorithms comprise of monotonic computations, hence error propagation of the application can be approximated to be

a monotonic function. The approach relies on binary-search to find the faults that lead to safety violations efficiently. The framework is compared to approaches that use random FI and shows significantly better results in terms of performance and cost, with 99.56% of the faults detected within 99.63% accuracy. The performance gain is 5 times more than random FI, however, the authors note that although the framework currently is not covering 100% of safety-critical faults, its accuracy is within a tolerable 0.5% range. Reference [37] argues that it is a fair trade-off for FI approaches to give up a small margin of accuracy to have efficiency advantages.

Reference [6] argues that BinFI has an advantage over DeepXplore proposed in reference [8], in a way that it covers significantly more errors, and additionally it also covers transient faults, while DeepXplore only considers systematic failures.

E. MUTATION TESTING

According to reference [4], while FI is an efficient way to detect faults in the system that violate safety goals, mutation testing is required to statistically prove that the system conforms to ISO 26262 safety requirements; FI can still be useful to identify test cases leading to system failures, thus reducing the testing space for mutation testing. Mutation testing is an approach used to evaluate the sufficiency of the test cases. The technique involves creating faulty duplicates of the original system, named mutants, by injecting reproducible faults. It allows engineers to have a metric that corresponds to the ratio of identified vs missed mutations, in order to evaluate the adequacy of test suites [4]. Mutants are identified if the output of the mutated software is different than that of the original one [58]. Reference [4] states that mutation testing is an efficient method to mock real faults. Mutation testing relies on two basic assumptions. One of them is the Competent Programmer Hypothesis, which is an assumption that the programmers develop software that is close to its perfect form. This assumption translates to mutation testing context as follows: mutants are close to real faults, and if the test suite is able to identify the mutants it will also be able to identify the real faults. Another assumption is the Coupling Effect hypothesis which states that large effects of the errors in the software are tightly coupled with small bugs. This assumption suggests that simple mutants are closely related to the real bugs in the software, and if it is possible to detect the mutants, it is also possible to detect critical bugs [4].

Reference [58] focuses on assessing the effectiveness of the current mutation tools for DNN. During the tests in the study, a threshold (1%) is set to check the deviations in the accuracy of decisions of DNN algorithms. Reference [58] demonstrates that it is possible to detect mutants with a deviation that exceeds the threshold and there was no case that no mutant was detected. To show the effectiveness of the tests a criterion called mutation score, which is the ratio of detected mutants to the remaining mutants, is used. Reference [58] notes that such criterion for the quality of test suits is important to show that safety-critical systems such as autonomous cars meet the Modified Condition/Decision Coverage criterion as recommended in ISO 26262. Experiments in the study show that the mutation score of experimented algorithms varies from 40% to 65%. The results also suggest that mutation operators related to learning behavior seem to have better performance than the others.

F. SOFTWARE SAFETY CAGES

A software safety cage is a safety mechanism that monitors the system behavior and performs appropriate actions if a malfunction is detected. Reference [68] suggests the usage of safety cages to reduce the strictness of requirements over NN-related applications in AVs. Since DNNs are usually seen as black boxes during the verification process, it is challenging to guarantee safety goals. However, using safety cages the behavior of the AV can be limited to a safe set of states. As an example, the decision of acceleration in a vehicle can be considered safe only if there is no vehicle in the front. Therefore, the environment should be monitored in real-time and, depending on the situation, system's limits should be controlled dynamically to always operate inside safety envelopes. The advantage of implementing safety cages is that the way they work is explainable, and conventional verification methods can be applied to them. They also do not require any knowledge about the inner workings of the ML system they are integrated with. The study in reference [68] focuses on safety cages that are created to avoid a forward collision. The experiments run in a simulation show that safety cages can effectively prevent collisions from happening. It is also demonstrated that the safety cages do not interfere with the system controllers if there is no safety-critical case, thus they do not reduce the performance of the vehicle under normal circumstances. Additionally, the authors suggest that identified safety cage violations can further be used to train and improve the network.

Another methodology to verify software systems in an iterative manner within the autonomous systems domain is proposed in reference [7]. The study introduces dependability cages to check situations where the system can behave outside its defined specifications in the development stage. In unexpected scenarios, dependability cages are able to interfere with the system configuration and collect information about these situations to provide them as feedback for further development. In this approach, similar to the monitor-actuator architecture, the functions or the system are deactivated to reach a safe state when unexpected events occur.

G. TECHNIQUES FOR CYBER-PHYSICAL SYSTEMS

In the context of AVs, Cyber-Physical Systems (CPSs) are defined as systems in which the physical behavior of the vehicle is controlled by computer-based algorithms without human intervention [39]. Although we have defined this separate category due to the number of studies specifically focusing on CPS V&V paradigms that can be transferred to/from multiple domains (e.g., rail transportation, manufacturing, etc.), the issues addressed by those studies largely

overlap with the ones already discussed in other categories. In fact, utilizing AI in CPSs creates new challenges in safety assessment due to the unpredictable nature of AI algorithms. Currently, CPS development relies on the assumption that all requirements for the system are complete and well-defined; however, considering the huge number of situations that might happen in the real world, the requirements are unlikely to cover all of them [7]. Therefore, new approaches should be adopted to assess the safety of smart and adaptive CPS.

The safety of conventional CPSs can be verified using formal methods [39] leveraging on the heritage of Real-Time and Embedded Systems research. The main challenges emerge when AI algorithms are involved. Extending the study in reference [28], reference [39] explores the ways of combining formal verification and AI together in CPSs. The study proposes an approach that uses differential dynamic logic to design CPS models with mathematical accuracy. Then ModelPlex method is used to guarantee that the verification results from the CPS model will be relevant for the actual implementation of the system. Lastly, the VeriPhy verification pipeline is used to ensure that safety verification results are preserved while the model changes by learning. It should be noted that, while safe control of CPSs can be proven using formal methods, it is under the assumption that sensors and actuators are working properly and provide correct information about the state of the system [39].

Reference [34] proposes an approach to formally verify the non-functional safety properties of CPSs. The study is then extended in reference [65] to verify both functional and non-functional properties using Simulink Design Verifier and UPPAAL-SMC for model checking. In reference [65], verification models are provided and explained step-by-step. A running example of a traffic sign recognition system is modeled and verified with 12 properties related to time and energy constraints.

Reference [20] investigates ways to improve the efficiency of testing CPSs in a simulation. The study focuses on finding a correlation between the modifications to CPS software and its decisions in order to reduce the number of test cases to be executed. The objective of the approach is to only choose tests cases that can show different results upon changes in the system. Experimenting with a lane-following algorithm shows that the approach could eliminate around 11% of the test cases, reducing the time required for testing [20]. Further research plan in reference [21] includes the use of namespace separation and virtual machines to reduce the execution time of testing in CPSs.

H. FORMAL METHODS

Testing is a the main approach used by engineers to find the inconsistencies and defects in the software, however, as mentioned earlier, testing can only prove the presence of errors, not their absence. In other words, testing can help to decrease the number of defects, but considered alone it is not sufficient to prove compliance against certification requirements [79]. With the aim of achieving both correctness and completeness, more formal approaches can and should be applied to AV software. For instance, according to reference [9], model-based testing is currently the most viable approach for conforming the quality of safety-critical systems such as autonomous cars.

More in general, together with fully formal approaches such as model-checking and theorem proving, which can be extremely challenging when dealing with complex and adaptive systems, several hybrid or semi-formal approaches have been proposed where formality actually refers to the methodology with which testing is performed. For instance, reference [79] proposes a formal methodology to verify the safety of embedded software in AVs. The proposed approach utilizes black-box testing techniques and can be used in real-time. This way the verification can be done online without human intervention. In the verification process, a model that conforms to the same requirements as the real system is designed. The same input is fed into both model (as a digital signal) and the system (as a physical signal), and their outputs are compared to generate a test report about whether the test cases failed or passed.

While a holistic approach is always needed in safety assessment, one way to manage the complexity and thus enable more extensive usage of formal methods is to divide the system into simpler components that can be isolated and checked by their own. Components in AVs can be divided into two categories in terms of safety, low-level and high-level. Low-level components are the ones for perception and actuation, while high-level components can be defined as the ones that are responsible for making decisions [9]. The study in reference [9] focuses on verifying the safe navigation of the vehicle considering obstacle avoidance and selection (if there is no way to avoid the obstacle). While formal verification can be applied to different software components of autonomous cars, the study uses the method to verify whether the vehicle makes correct decisions. LTL (Linear Temporal Logic) formulas that represent the desired decisions and state of the system are defined as properties and checked by the AJPF tool. However, the authors note that the approach does not yet consider real-world scenarios in depth, and future improvements are necessary.

Formal methods are based on sound mathematical models and hence are very appropriate in the V&V of safe AVs. However, one essential requirement is that the models being checked should conform to the real systems. Although carefully designed models may hold the properties of interest, it may happen that they are not always complete. It is particularly difficult and expensive to build CPSs that are complete in the context of AVs, as these systems are expected to function in open environments which make inconsistencies between the model and reality unavoidable. For instance, some reinforcement learning approaches can be applied without reference models, and, although the learning algorithms are effective, their safe behavior cannot be proven [27]. Therefore, reference [27] proposes a technique combining two approaches: optimized learning algorithms with high

TABLE 3. Tools supporting the software V&V of safe autonomous cars.

Tool Name	Availability	Application	User interaction	White-box testing	Mutation testing	Fault injection	Simulation	Goal
AsFault	GitHub	AVs	Command line, Graph. interface			x		Automatic generation of virtual tests for systematic software testing of AVs
AVFI	GitHub	AVs	Python			x		Introduce faults to test AV resilience in rare situations
BinFI	GitHub	General	Python			x		Find safety-critical bits in ML applications
DeepSmart Fuzzer	GitHub	General	Python		x			Find coverage-guided fuzzing solution for structural testing of DNN
DeepXplore	GitHub	General	Python	x				White-box testing of DL models
DLFuzz	GitHub	General	Python	x				Generate tests to maximize difference between original and mutated inputs
MOBATSim	GitHub, Website	AVs	MATLAB, Simulink			x		Assess safety of vehicles and traffic as a whole
OpenPilot (agent)	GitHub	AVs	Graphical interface			x		Assess resilience of open-source driving agents
SHARC	Website	General	UML profile				x	Simulation and verification framework
TensorFI	GitHub	General	Python			x		Evaluate resilience of ML applications

exploration capabilities are coupled with formal verification methods that can ensure system safety. The main objective of the proposed approach is to verify learning in run-time. As long as the reinforcement learning algorithm does not violate the requirements, the learning process continues efficiently and the verification results are saved. When a violation occurs, the efficient way of learning gives its place to a different algorithm that tries to reach the states that conform to the model and requirements. Reference [27] compares the approach to other state-of-the-art approaches and shows its computational advantages as it performs verification during learning.

In complex systems such as AVs, problems can also arise considering the interaction between hardware and software. Amongst other things, the software system needs to recognize and manage transient faults in the hardware. To that aim, a standard architecture called AUTOSAR is used in the automotive industry. Within this architecture, software applications can be implemented on several electronic control units (ECUs) [82]. In such a context, reference [82] proposes a verification approach for software systems running on hardware platforms subject to transient faults. To be able to formally verify the model, the AUTOSAR model is converted into timed automata. The method has been implemented and experimented on an autonomous car and the effectiveness of the approach has been demonstrated.

Table 3 summarizes the main tools supporting the software V&V of safe autonomous cars, as emerged from the review reported in this section. Most of them are available on the GitHub repository (www.github.com) and support fault injection approaches.

V. OPEN ISSUES, CHALLENGES AND OPPORTUNITIES

The SLR performed in this paper has highlighted a rising interest in the research area of software V&V for safe autonomous cars. In this section, the main open issues and challenges are pointed out and promising opportunities in the V&V of safe autonomous cars are outlined to support future research directions.

The biggest challenge in the V&V of safe autonomous cars is due to the fact that achieving autonomy requires using complex ML algorithms that need to be integrated into the control software. While these algorithms are capable of providing highly accurate results, their behavior is generally unpredictable. The decision-making process of some algorithms may be explainable in human terms, however, how to make these algorithms ''legible'' for humans is still an open issue [3].

Specific black-box testing approaches have been demonstrated to be a viable option to test ML algorithms against adversarial examples. The most effective black-box testing approach for DNNs mentioned in this SLR is the one presented in reference [47]. The authors have experimented their approach on two state-of-the-art networks and concluded that they have not yet found a network that was safe in all conditions. This result suggests that current DNN solutions are not mature enough to be relied on in safety-critical systems such as autonomous cars.

Two fuzzy testing frameworks, DLFuzz [31] and DeepSmartFuzzer [55] both showed high performance and accuracy in detecting adversarial examples. However, their effectiveness cannot be only measured in terms of achieved neuron coverage, since it is noted in references [8], [58] that 100% neuron coverage can easily be achieved using certain input vectors from the training set. The approach shown in reference [38] does not rely on neuron coverage and can be an efficient alternative with a very promising 96.85% accuracy. Reference [38] suggests further research and experiments (with popular datasets such as MNIST) on using affine transformations to find adversarial examples. Although there are extensive research efforts on how to find adversarial examples in DNNs, how to avoid these attacks is still an open issue [24].

Furthermore, the performance of ML algorithms hugely depends on the quality of training data. As mentioned in Section III, the training data should be diverse and the

resulting model should not suffer from over-fitting. However, there is currently no general way to either assess if the data is sufficient in terms of quantity and diversity, or prove that the model is not over-fitted. Reference [3] argues that even creating a set of requirements for collecting the data will not cause any progress towards the solution, since now the adequacy of the requirements would need validation in the same way, merely shifting the problem one layer up.

While being valuable, conventional testing methods are limited in terms of completeness and that is especially true in the autonomous cars domain [3]. Formal methods are demonstrated to be a viable option to achieve completeness, however they do not scale up well to complex system and they still rely on the correctness of sensors and perception algorithms. Reference [39] mentions that even if sensors can be wrong, their output can still be partly considered and it is thus necessary to establish a limit for sensor errors, which is an open issue. Additionally, how formal properties for perception-related algorithms can be defined is so far an unanswered question [28].

One promising option to consider unreliable event detection in combination with sensor redundancy, diversity and self-adaptation techniques, is to adopt probabilistic XAI approaches based on Bayesian Networks, enabling run-time ASIL estimation and assurance against quantitative safety targets such as the hazard rate [203].

As mentioned in reference [79], one concern in model-based formal verification is the assumption that the model being checked reflects the properties of the actual system. Similarly, reference [28] mentions that it is possible to ensure the safety of learning algorithms using formal properties only if the environmental model is correct. Reference [28] suggests that future research directions in this area will be towards ensuring safety when the state or action space is continuous and when the environmental model is not available or is not accurately defined.

Regarding mitigation of transient faults in AVs, we have mentioned formal approaches for CPSs [82] and fault injection (BinFI [6]). In particular, BinFI does not only cover transient errors but also systematic errors with better error coverage than DeepXplore [8]. To further demonstrate the performance of BinFI, it might be useful to consider making a comparison between BinFI and DLFuzz presented in reference [31], as DLFuzz is shown to perform better than DeepXplore.

Regarding simulation-based approaches, developments in MOBATSim [75] should be seriously considered due to its potential to become an advanced safety analysis tool for AVs. Authors plan to add a report generation feature conforming to ISO 26262 specifications, which might be valuable for the industry. Additionally, a recent prototype, AsFault, presented in reference [30] have been demonstrated to be an effective approach to automatically generate test cases, and future works in that direction seem very promising.

Another research direction to consider is the usage of safety cages which might help to reduce the assessment burden of ML algorithms. Since these algorithms are challenging to assess, safety cages can be used to limit the behavior of the vehicle to a safe envelope, avoiding hazardous scenarios from happening due to unexpected conditions. In this way, even if the system cannot be completely verified, a safe operation can be achieved. Furthermore, reference [68] suggests utilizing safety cages not only to achieve safe operation but also to find corner cases in ML algorithms, and that seems another promising opportunity.

It is important to underline that there are several other aspects connected to human factors, ethics, and the safety of AVs in combination with specific SAE levels, which are being currently investigated by the research community, and that could possibly have an impact on future evolution of reference standards as well as on the software design for driver-machine interfaces (DMIs); however, all the open issues, challenges and opportunities related to those important aspects of safe AVs, but not specifically to their software V&V, were not in the scope of this SLR.

VI. RELATED WORK

A parallel SLR on testing and verification of NN-based safety-critical control software has been recently performed and published in reference [204]. The review covered 83 studies published between 2011 and 2018, and mainly focused on NN and CPSs. Although the review covered several important papers in the safety-critical systems domain, only 13 of them were connected to the automotive field. Since ISO 26262 is an adaptation of the IEC 61508 standard with a specific focus on on-road vehicles, reference [204] adopts IEC 61508 as a reference standard rather than ISO 26262. Therefore, papers selected in reference [204] do not completely represent the domain considered in this SLR due to different search criteria and topic coverage. Reference [205] very recently provided a useful overview of the factors influencing the adoption of AVs, including safety challenges. A total of 14 factors out of 85 articles have been highlighted in that study. Compared to the study performed in this paper, reference [205] only focused on identifying the general industry challenges without addressing technical details and the state-of-the-art of relevant research. A valuable systematic literature review on the specific topic of coverage-based testing for self-driving autonomous vehicles is presented in reference [88]. The review classifies available literature based on coverage-criteria used in the studies. Out of 89 studies related to V&V of AVs, 26 studies that use coverage-based testing are selected after applying inclusion and exclusion criteria. One important conclusion of the study is that the terminology of coverage criteria used in V&V processes is not standardized throughout the literature. Therefore, the authors suggest unification of terminologies to increase the progress pace of the research in that field. We have addressed testing coverage criteria in Section IV-C.

VII. CONCLUSION

This paper provided a systematic literature review on software V&V of safe autonomous cars, covering the research work in - approximately - the last ten years. Three research questions have been defined to provide an in-depth analysis of current challenges, state-of-the-art and future research directions:

- RQ1: What are the most common requirements in the software V&V of safe autonomous cars?
- RQ2: What are the main challenges in performing software V&V of safe autonomous cars?
- RQ3: What are the open issues and opportunities in software V&V of safe autonomous cars?

In order to answer those questions, we have identified the main V&V requirements according to safety standards such as the ISO 26262, and provided a summary of most important certification objectives (RQ1). Furthermore, we have classified the main approaches available today to check that software in AVs meets given safety requirements, together with a comparison of shortcomings and potential of relevant methods (RQ2). Finally, we have investigated the state-of-the-art in structured categories and described open issues, future opportunities and directions (RQ3).

In total, more than 200 relevant papers have been investigated in the SLR, and more than half of them have been further selected as a base for discussion in this study. Throughout the study, it has been observed that the current trend in the V&V of safe autonomous cars is towards extending already existing standards and processes such as the ISO 26262 and its V-model. Likewise, the progress in V&V processes is mostly achieved by adapting and integrating diverse approaches such as formal verification and fault injection, also considering novel techniques needed to tackle the emerging challenges of smart-CPS paradigms. One somehow expected result of the study is that although state-of-the-art approaches seem promising, they are currently inadequate to ensure the safety of fully autonomous cars in all operating conditions. In the software part, this is primarily due to the lack of approaches to make ML algorithms fully explainable and predictable, also considering adversarial attacks and training data validation issues. Nevertheless, the usage of safety/dependability cages as well as monitor and actuator approaches have demonstrated a great potential to compensate for incomplete or immature V&V processes.

In conclusion, we hope that due to its quite extensive topic coverage, this paper can serve as a useful compendium for the many engineers and researchers who are starting to investigate those extremely current and challenging subjects related to the software safety of autonomous road vehicles. Although V&V for AVs is a less mature area compared to other safety-critical domains, we believe that the pioneering research and rapid progress in this sector connected to the need for quickly reaching higher levels of autonomy, pushed by private investments, can also be very useful in the perspective of technology transfer towards other transport sectors such as smart railways.

REFERENCES

[1] N. Navet and F. Simonot-Lion, *Automation Embedded System Handbook*. Boca Raton, FL, USA: CRC Press, 2017.
[2] S. Chen, Y. Leng, and S. Labi, "A deep learning algorithm for simulating autonomous driving considering prior knowledge and temporal information," *Comput.-Aided Civil Infrastruct. Eng.*, vol. 35, no. 4, pp. 305–321, Apr. 2020.
[3] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *SAE Int. J. Transp. Saf.*, vol. 4, no. 1, pp. 15–24, Apr. 2016.
[4] R. Rana, M. Staron, C. Berger, J. Hansson, M. Nilsson, and F. Törner, "Early verification and validation according to iso 26262 by combining fault injection and mutation testing," *Commun. Comput. Inf. Sci.*, vol. 457, pp. 164–179, Jul. 2013.
[5] A. Takacs, D. A. Drexler, P. Galambos, I. J. Rudas, and T. Haidegger, "The transition of L2–L3 autonomy through euro NCAP highway assist scenarios," in *Proc. IEEE 17th World Symp. Appl. Mach. Intell. Informat. (SAMI)*, Jan. 2019, pp. 117–122.
[6] Z. Chen, G. Li, K. Pattabiraman, and N. DeBardeleben, "BinFI: An efficient fault injector for safety-critical machine learning systems," in *Proc. Int. Conf. for High Perform. Comput., Netw., Storage Anal.*, Nov. 2019, pp. 1–23.
[7] A. Aniculaesei, J. Grieser, A. Rausch, K. Rehfeldt, and T. Warnecke, "Towards a holistic software systems engineering approach for dependable autonomous systems," in *Proc. 1st Int. Workshop Softw. Eng. Auto. Syst.*, May 2018, pp. 23–30.
[8] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," in *Proc. 26th Symp. Oper. Syst. Princ.*, 2017, pp. 1–18.
[9] P. Kaur and R. Sobti, "Current challenges in modelling advanced driver assistance systems: Future trends and advancements," in *Proc. 2nd IEEE Int. Conf. Intell. Transp. Eng.*, Sep. 2017, pp. 236–240.
[10] R. Weissnegger, C. Kreiner, M. Pistauer, K. Römer, and C. Steger, *SHARC-Simulation and Verification of Hierarchical Embedded Microelectronic Systems*, vol. 109. Amsterdam, The Netherlands: Elsevier, 2017, pp. 392–399.
[11] F. Flammini, Z. Lin, and V. Vittorini, "Roadmaps for ai integration in the rail sector—Rails," *ERCIM News*, vol. 2020, pp. 34–35, Apr. 2020.
[12] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," *J. Syst. Softw.*, vol. 80, no. 4, pp. 571–583, Apr. 2007.
[13] S. Keele, "Guidelines for performing systematic literature reviews in software engineering," EBSE, Goyang-Si, South Korea, Tech. Rep. 2.3, 2007.
[14] X. Xie, L. Ma, F. Juefei-Xu, M. Xue, H. Chen, Y. Liu, J. Zhao, B. Li, J. Yin, and S. See, "DeepHunter: A coverage-guided fuzz testing framework for deep neural networks," in *Proc. 28th ACM SIGSOFT Int. Symp. Softw. Test. Anal.*, Jul. 2019, pp. 158–168.
[15] S. Jha, S. Banerjee, T. Tsai, S. K. S. Hari, M. B. Sullivan, Z. T. Kalbarczyk, S. W. Keckler, and R. K. Iyer, "ML-based fault injection for autonomous vehicles: A case for Bayesian fault injection," in *Proc. 49th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2019, pp. 112–124.
[16] B. Xu, Q. Li, T. Guo, Y. Ao, and D. Du, "A quantitative safety verification approach for the decision-making process of autonomous driving," in *Proc. Int. Symp. Theor. Aspects Softw. Eng.*, Jul. 2019, pp. 128–135.
[17] B. Gangopadhyay, S. Khastgir, S. Dey, P. Dasgupta, G. Montana, and P. Jennings, "Identification of test cases for automated driving systems using Bayesian optimization," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 1961–1967.
[18] H. Abbas, I. Saha, Y. Shoukry, R. Ehlers, G. Fainekos, R. Gupta, R. Majumdar, and D. Ulus, "Special session: Embedded software for robotics: Challenges and future directions," in *Proc. Int. Conf. Embedded Softw. (EMSOFT)*, Sep. 2018, pp. 1–10.
[19] R. B. Abdessalem, A. Panichella, S. Nejati, L. C. Briand, and T. Stifter, "Testing autonomous cars for feature interaction failures using many-objective search," in *Proc. 33rd ACM/IEEE Int. Conf. Automated Softw. Eng.*, Sep. 2018, pp. 143–154.
[20] C. Berger, "Saving virtual testing time for CPS by analyzing code coverage on the example of a lane-following algorithm," in *Proc. 4th ACM SIGBED Int. Workshop Design, Modeling, Eval. Cyber-Phys. Syst.*, 2014, pp. 7–10.

[21] C. Berger, "Accelerating regression testing for scaled self-driving cars with lightweight virtualization—A case study," in *Proc. IEEE/ACM 1st Int. Workshop Softw. Eng. Smart Cyber-Phys. Syst.*, May 2015, pp. 2–7.

[22] H. B. Braiek and F. Khomh, "TFCheck: A tensorflow library for detecting training issues in neural network programs," in *Proc. IEEE 19th Int. Conf. Softw. Qual., Rel. Secur. (QRS)*, Jul. 2019, pp. 426–433.

[23] T. Byun, V. Sharma, A. Vijayakumar, S. Rayadurgam, and D. Cofer, "Input prioritization for testing neural networks," in *Proc. IEEE Int. Conf. Artif. Intell. Test. (AITest)*, Apr. 2019, pp. 63–70.

[24] A. Chernikova, A. Oprea, C. Nita-Rotaru, and B. Kim, "Are self-driving cars secure? Evasion attacks against deep neural networks for steering angle prediction," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2019, pp. 132–137.

[25] T. Fehlmann and E. Kranich, "Autonomous real-time software & systems testing," in *Proc. 12th Int. Conf. Softw. Process Product Meas.*, Oct. 2017, pp. 54–63.

[26] T. Fehlmann, "Testing artificial intelligence," *Commun. Comput. Inf. Sci.*, vol. 1060, pp. 709–721, May 2019.

[27] N. Fulton and A. Platzer, "Safe reinforcement learning via formal methods: Toward safe control through proof and learning," in *Proc. AAAI*, 2018, pp. 6485–6492.

[28] N. Fulton and A. Platzer, "Safe ai for CPS (invited paper)," in *Proc. IEEE Int. Test Conf.*, Oct. 2018, pp. 1–7.

[29] A. Gambi, M. Müller, and G. Fraser, "AsFault: Testing self-driving car software using search-based procedural content generation," in *Proc. IEEE/ACM 41st Int. Conf. Softw. Eng., Companion Proc. (ICSE-Companion)*, May 2019, pp. 27–30.

[30] A. Gambi, M. Müller, and G. Fraser, "Automatically testing self-driving cars with search-based procedural content generation," in *Proc. 28th ACM SIGSOFT Int. Symp. Softw. Test. Anal.*, Jul. 2019, pp. 273–283.

[31] J. Guo, Y. Jiang, Y. Zhao, Q. Chen, and J. Sun, "DLFuzz: Differential fuzzing testing of deep learning systems," in *Proc. 26th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, Oct. 2018, pp. 739–743.

[32] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," in *Computer Aided Verification* (Lecture Notes in Computer Science), vol. 10426. Cham, Switzerland: Springer, 2017, pp. 3–29.

[33] R. Hussain and S. Zeadally, "Autonomous cars: Research results, issues, and future challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1275–1313, 2nd Quart., 2019.

[34] E.-Y. Kang, D. Mu, L. Huang, and Q. Lan, "Model-based analysis of timing and energy constraints in an autonomous vehicle system," in *Proc. IEEE Int. Conf. Softw. Qual., Rel. Secur. Companion (QRS-C)*, Jul. 2017, pp. 525–532.

[35] J. Kim, S. Chon, and J. Park, "Suggestion of testing method for industrial level cyber-physical system in complex environment," in *Proc. IEEE Int. Conf. Softw. Test., Verification Validation Workshops (ICSTW)*, Apr. 2019, pp. 148–152.

[36] F. Kluck, M. Zimmermann, F. Wotawa, and M. Nica, "Genetic algorithm-based test parameter optimization for ADAS system testing," in *Proc. IEEE 19th Int. Conf. Softw. Qual., Rel. Secur. (QRS)*, Jul. 2019, pp. 418–425.

[37] G. Li, K. Pattabiraman, and N. DeBardeleben, "TensorFI: A configurable fault injector for tensorflow applications," in *Proc. IEEE Int. Symp. Softw. Rel. Eng. Workshops (ISSREW)*, Oct. 2018, pp. 313–320.

[38] R. R. Mekala, G. E. Magnusson, A. Porter, M. Lindvall, and M. Diep, "Metamorphic detection of adversarial examples in deep learning models with affine transformations," in *Proc. IEEE/ACM 4th Int. Workshop Metamorphic Test. (MET)*, May 2019, pp. 55–62.

[39] A. Platzer, "The logical path to autonomous cyber-physical systems," *Quantitative Evaluation of Systems* (Lecture Notes in Computer Science), vol. 11785. Cham, Switzerland: Springer, 2019, pp. 25–33, 2019.

[40] R. Queiroz, T. Berger, and K. Czarnecki, "GeoScenario: An open DSL for autonomous driving scenario representation," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2019, pp. 287–294.

[41] M. S. Ramanagopal, C. Anderson, R. Vasudevan, and M. Johnson-Roberson, "Failing to learn: Autonomously identifying perception failures for self-driving cars," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3860–3867, Oct. 2018.

[42] E. Rocklage, "Teaching self-driving cars to dream: A deeply integrated, innovative approach for solving the autonomous vehicle validation problem," in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2017, pp. 1–7.

[43] H. Tabani, L. Kosmidis, J. Abella, F. J. Cazorla, and G. Bernat, "Assessing the adherence of an industrial autonomous driving framework to ISO 26262 software guidelines," in *Proc. 56th Annu. Design Autom. Conf.*, Jun. 2019, pp. 1–6.

[44] E. R. Teoh and D. G. Kidd, "Rage against the machine? Google's self-driving cars versus human drivers," *J. Saf. Res.*, vol. 63, pp. 57–60, Dec. 2017.

[45] H. J. Vishnukumar, B. Butting, C. Müller, and E. Sax, "Machine learning and deep neural network—Artificial intelligence core for lab and real-world test and validation for ADAS and autonomous vehicles: AI for efficient and quality test and validation," in *Proc. Intell. Syst. Conf. (IntelliSys)*, Sep. 2017, pp. 714–721.

[46] A. von Bernuth, G. Volk, and O. Bringmann, "Rendering physically correct raindrops on windshields for robustness verification of camera-based object recognition," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2018, pp. 922–927.

[47] M. Wicker, X. Huang, and M. Kwiatkowska, "Feature-guided black-box safety testing of deep neural networks," in *Tools and Algorithms for the Construction and Analysis of Systems* (Lecture Notes in Computer Science), vol. 10805. Cham, Switzerland: Springer, 2018, pp. 408–426.

[48] M. Wu, M. Wicker, W. Ruan, X. Huang, and M. Kwiatkowska, "A game-based approximate verification of deep neural networks with provable guarantees," *Theor. Comput. Sci.*, vol. 807, pp. 298–329, Feb. 2020.

[49] 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 *Proc. 33rd ACM/IEEE Int. Conf. Autom. Softw. Eng.*, Sep. 2018, pp. 132–142.

[50] T. Laurent, P. Arcaini, F. Ishikawa, and A. Ventresque, "A mutation-based approach for assessing weight coverage of a path planner," in *Proc. 26th Asia–Pacific Softw. Eng. Conf. (APSEC)*, Dec. 2019, pp. 94–101.

[51] Y. Li, J. Tao, and F. Wotawa, "Ontology-based test generation for automated and autonomous driving functions," *Inf. Softw. Technol.*, vol. 117, Jan. 2020, Art. no. 106200.

[52] F. U. Haq, D. Shin, S. Nejati, and L. C. Briand, "Comparing offline and online testing of deep neural networks: An autonomous car case study," in *Proc. IEEE 13th Int. Conf. Softw. Test., Validation Verification (ICST)*, Oct. 2020, pp. 85–95.

[53] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, and X. Yi, "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability," *Comput. Sci. Rev.*, vol. 37, Aug. 2020, Art. no. 100270.

[54] J. Jung, M. Park, K. Cho, C. Mun, and J. Ahn, "Intelligent hybrid fusion algorithm with vision patterns for generation of precise digital road maps in self-driving vehicles," *KSII Trans. Internet Inf. Syst.*, vol. 14, no. 10, pp. 3955–3971, 2020.

[55] S. Demir, H. Eniser, and A. Sen, *Deepsmartfuzzer: Reward Guided Test Generation for Deep Learning*, vol. 2640, M. R. Espinoza and H. McDermid, Eds. Anissaras, Greece: CEUR-WS, 2020.

[56] A. Bhat, S. Aoki, and R. Rajkumar, "Tools and methodologies for autonomous driving systems," *Proc. IEEE*, vol. 106, no. 9, pp. 1700–1716, Sep. 2018.

[57] C.-H. Cheng, F. Diehl, G. Hinz, Y. Hamza, G. Nuehrenberg, M. Rickert, H. Ruess, and M. Truong-Le, "Neural networks for safety-critical applications—Challenges, experiments and perspectives," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 1005–1006.

[58] N. Chetouane, L. Klampfl, and F. Wotawa, "Investigating the effectiveness of mutation testing tools in the context of deep neural networks," in *Advances in Computational Intelligence* (Lecture Notes in Computer Science), vol. 11506. Cham, Switzerland: Springer, 2019, pp. 766–777.

[59] T. Denewiler and M. Tjersland, "Best practices for autonomous vehicle configuration management," *Proc. SPIE Unmanned Syst. Technol.*, vol. 10195, May 2017, Art. no. 101950N.

[60] T. Duy Son, A. Bhave, and H. Van der Auweraer, "Simulation-based testing framework for autonomous driving development," in *Proc. IEEE Int. Conf. Mechatronics (ICM)*, Mar. 2019, pp. 576–583.

[61] L. E. R. Fernandes, V. Custodio, G. V. Alves, and M. Fisher, "A rational agent controlling an autonomous vehicle: Implementation and formal verification," 2017, *arXiv:1709.02557*. [Online]. Available: https://arxiv.org/abs/1709.02557

[62] Y. Fu, A. Terechko, T. Bijlsma, P. J. L. Cuijpers, J. Redegeld, and A. O. Ors, "A retargetable fault injection framework for safety validation of autonomous vehicles," in *Proc. IEEE Int. Conf. Softw. Archit. Companion (ICSA-C)*, Mar. 2019, pp. 69–76.

[63] J. Gustavsson, "Verification methodology for fully autonomous heavy vehicles," in *Proc. IEEE Int. Conf. Softw. Test., Verification Validation (ICST)*, Apr. 2016, pp. 381–382.
[64] S. Jha, S. S. Banerjee, J. Cyriac, Z. T. Kalbarczyk, and R. K. Iyer, "AVFI: Fault injection for autonomous vehicles," in *Proc. 48th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. Workshops (DSN-W)*, Jun. 2018, pp. 55–56.
[65] E.-Y. Kang, D. Mu, L. Huang, and Q. Lan, "Verification and validation of a cyber-physical system in the automotive domain," in *Proc. IEEE Int. Conf. Softw. Qual., Rel. Secur. Companion (QRS-C)*, Jul. 2017, pp. 326–333.
[66] B. Kim, Y. Kashiba, S. Dai, and S. Shiraishi, "Testing autonomous vehicle software in the virtual prototyping environment," *IEEE Embedded Syst. Lett.*, vol. 9, no. 1, pp. 5–8, Mar. 2017.
[67] F. Klăck, M. Zimmermann, F. Wotawa, and M. Nica, "Performance comparison of two search-based testing strategies for adas system validation," in *Testing Software and System* (Lecture Notes in Computer Science), vol. 11812. Berlin, Germany: Springer, 2019, pp. 140–156.
[68] S. Kuutti, R. Bowden, H. Joshi, R. de Temple, and S. Fallah, "Safe deep neural network-driven autonomous vehicles using software safety cages," in *Intelligent Data Engineering and Automated Learning* (Lecture Notes in Computer Science), vol. 11872. Cham, Switzerland: Springer, 2019, pp. 150–160.
[69] D. Meltz and H. Guterman, "RobIL—Israeli program for research and development of autonomous UGV: Performance evaluation methodology," in *Proc. IEEE Int. Conf. Sci. Electr. Eng. (ICSEE)*, Nov. 2016, pp. 1–5.
[70] F. Narisawa and Y. Ueda, "Safety verification method for priority-based real-time software," *Frontiers Artif. Intell. Appl.*, vol. 297, pp. 409–424, Dec. 2017.
[71] D. Negrut, R. Serban, A. Elmquist, D. Hatch, E. Nutt, and P. Sheets, "Autonomous vehicles in the cyberspace: Accelerating testing via computer simulation," SAE Tech. Papers 2018-01-1078, Apr. 2018.
[72] A. C. Nica, M. Trascau, A. A. Rotaru, C. Andreescu, A. Sorici, A. M. Florea, and V. Bacue, "Collecting and processing a self-driving dataset in the UPB campus," in *Proc. 22nd Int. Conf. Control Syst. Comput. Sci. (CSCS)*, May 2019, pp. 202–209.
[73] D. Rao, P. Pathrose, F. Huening, and J. Sid, "An approach for validating safety of perception software in autonomous driving systems," in *Model-Based Safety and Assessment* (Lecture Notes in Computer Science), vol. 11842. Cham, Switzerland: Springer, 2019, pp. 303–316.
[74] A. H. M. Rubaiyat, Y. Qin, and H. Alemzadeh, "Experimental resilience assessment of an open-source driving agent," in *Proc. IEEE 23rd Pacific Rim Int. Symp. Dependable Comput. (PRDC)*, Dec. 2018, pp. 54–63.
[75] M. Saraoglu, A. Morozov, and K. Janschek, *Mobatsim: Model-Based Autonomous Traffic Simulation Framework for Fault-Error-Failure Chain Analysis*, vol. 52, no. 8, D. M. Wiszniewski and B. Kowalczuk, Eds. Amsterdam, The Netherlands: Elsevier, 2019, pp. 397–402.
[76] H. R. Schmidtke, "A survey on verification strategies for intelligent transportation systems," *J. Reliable Intell. Environ.*, vol. 4, no. 4, pp. 211–224, Dec. 2018.
[77] W. Shi, M. B. Alawieh, X. Li, H. Yu, N. Arechiga, and N. Tomatsu, "Efficient statistical validation of machine learning systems for autonomous driving," in *Proc. 35th Int. Conf. Comput.-Aided Design*, Nov. 2016, pp. 1–8.
[78] J. Sini, A. Mugoni, M. Violante, A. Quario, C. Argiri, and F. Fusetti, "An automatic approach to integration testing for critical automotive software," in *Proc. 13th Int. Conf. Design Technol. Integr. Syst. Nanosc. Era (DTIS)*, Apr. 2018, pp. 1–2.
[79] P. Skruch, M. DÅugosz, and P. Markiewicz, "A formal approach for the verification of control systems in autonomous driving applications," *Adv. Intell. Syst. Comput.*, vol. 577, pp. 178–189, Dec. 2017.
[80] C. Sun, L. Su, S. Gu, J. M. Uwabeza Vianney, K. Qin, and D. Cao, "Cross validation for CNN based affordance learning and control for autonomous driving," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 1519–1524.
[81] C. Wolschke, T. Kuhn, D. Rombach, and P. Liggesmeyer, "Observation based creation of minimal test suites for autonomous vehicles," in *Proc. IEEE Int. Symp. Softw. Rel. Eng. Workshops (ISSREW)*, Oct. 2017, pp. 294–301.
[82] R. Yan, J. Yang, D. Zhu, and K. Huang, "Design verification and validation for reliable safety-critical autonomous control systems," in *Proc. 23rd Int. Conf. Eng. Complex Comput. Syst. (ICECCS)*, Dec. 2018, pp. 170–179.
[83] S. Yoo, "SBST in the age of machine learning systems–challenges ahead," in *Proc. IEEE/ACM 12th Int. Workshop Search-Based Softw. Test. (SBST)*, May 2019, pp. 29–32.
[84] T. Winkle, C. Erbsmehl, and K. Bengler, "Area-wide real-world test scenarios of poor visibility for safe development of automated vehicles," *Eur. Transp. Res. Rev.*, vol. 10, no. 2, pp. 1–5, Jun. 2018.
[85] V. Remeli, S. Morapitiye, A. Rovid, and Z. Szalay, "Towards verifiable specifications for neural networks in autonomous driving," in *Proc. IEEE 19th Int. Symp. Comput. Intell. Informat.*, Nov. 2019, pp. 175–180.
[86] H. Ben Braiek and F. Khomh, "DeepEvolution: A search-based testing approach for deep neural networks," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Sep. 2019, pp. 454–458.
[87] D. An, J. Liu, M. Zhang, X. Chen, M. Chen, and H. Sun, "Uncertainty modeling and runtime verification for autonomous vehicles driving control: A machine learning-based approach," *J. Syst. Softw.*, vol. 167, Sep. 2020, Art. no. 110617.
[88] Z. Tahir and R. Alexander, "Coverage based testing for V&V and safety assurance of self-driving autonomous vehicles: A systematic literature review," in *Proc. IEEE Int. Conf. Artif. Intell. Test. (AITest)*, Aug. 2020, pp. 23–30.
[89] A. Stocco, M. Weiss, M. Calzana, and P. Tonella, "Misbehaviour prediction for autonomous driving systems," in *Proc. Conf. Comput. Soc.*, 2020, pp. 359–371.
[90] R. Ivanov, T. Carpenter, J. Weimer, R. Alur, G. Pappas, and I. Lee, "Case study: Verifying the safety of an autonomous racing car with a neural network controller," in *Proc. 23rd Int. Conf. Hybrid Syst., Comput. Control*, 2020, pp. 1–7.
[91] H. Alghodhaifi and S. Lakshmanan, "Simulation-based model for surrogate safety measures analysis in automated vehicle-pedestrian conflict on an urban environment," *Proc. SPIE Auto. Syst., Sensors, Process., Secur. Vehicles Infrastruct.*, vol. 11415, May 2020, Art. no. 1141504.
[92] B. Vedder, B. J. Svensson, J. Vinter, and M. Jonsson, "Automated testing of ultrawideband positioning for autonomous driving," *J. Robot.*, vol. 2020, pp. 1–15, Jan. 2020.
[93] Y. Li, M. Li, B. Luo, Y. Tian, and Q. Xu, "Deepdyve: Dynamic verification for deep neural networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2020, pp. 101–112.
[94] K. Pattabiraman, G. Li, and Z. Chen, "Error resilient machine learning for safety-critical systems: Position paper," in *Proc. IEEE 26th Int. Symp. On-Line Test. Robust Syst. Design (IOLTS)*, Jul. 2020, pp. 1–4.
[95] J. C. Han and Z. Q. Zhou, "Metamorphic fuzz testing of autonomous vehicles," in *Proc. IEEE/ACM 42nd Int. Conf. Softw. Eng. Workshops*, Jun. 2020, pp. 380–385.
[96] S. Kuutti, S. Fallah, and R. Bowden, "Training adversarial agents to exploit weaknesses in deep control policies," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 108–114.
[97] T. Bijlsma, A. Buriachevskyi, A. Frigerio, Y. Fu, K. Goossens, A. O. Ors, P. J. van der Perk, A. Terechko, and B. Vermeulen, "A distributed safety mechanism using middleware and hypervisors for autonomous vehicles," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2020, pp. 1175–1180.
[98] C.-H. Cheng, C.-H. Huang, T. Brunner, and V. Hashemi, "Towards safety verification of direct perception neural networks," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2020, pp. 1640–1643.
[99] M. Khaled and M. Zamani, "Cloud-ready acceleration of formal method techniques for cyber-physical systems," *IEEE Des. Test.*, early access, Oct. 26, 2020, doi: 10.1109/MDAT.2020.3034048.
[100] S. Manivasagam, S. Wang, K. Wong, W. Zeng, M. Sazanovich, S. Tan, B. Yang, W.-C. Ma, and R. Urtasun, "LiDARsim: Realistic LiDAR simulation by leveraging the real world," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, p. 11.
[101] A. Rössig and M. Petkovic, "Advances in verification of ReLU neural networks," *J. Global Optim.*, Oct. 2020, doi: 10.1007/s10898-020-00949-1.
[102] T. Wu, Y. Dong, Z. Dong, A. Singa, X. Chen, and Y. Zhang, "Testing artificial intelligence system towards safety and robustness: State of the art," *IAENG Int. J. Comput. Sci.*, vol. 47, no. 3, pp. 449–462, 2020.
[103] D. Garrido, L. Ferreira, J. Jacob, and D. Silva, "Fault injection, detection and treatment in simulated autonomous vehicles," in *Computational Science* (Lecture Notes in Computer Science), vol. 12137. Cham, Switzerland: Springer, 2020, pp. 471–485.
[104] S. Mohseni, M. Pitale, V. Singh, and Z. Wang, "Practical solutions for machine learning safety in autonomous vehicles," 2019, *arXiv:1912.09630*. [Online]. Available: https://arxiv.org/abs/1912.09630

[105] T. Wu, Y. Dong, Y. Zhang, and A. Singa, "ExtendAIST: Exploring the space of ai-in-the-loop system testing," *Appl. Sci.*, vol. 10, no. 2, p. 518, Jan. 2020.

[106] A. AbdelHamed, G. Tewolde, and J. Kwon, "Simulation framework for development and testing of autonomous vehicles," in *Proc. IEEE Int. IoT, Electron. Mechatronics Conf. (IEMTRONICS)*, Sep. 2020, pp. 471–485.

[107] A. Ruospo, A. Bosio, A. Ianne, and E. Sanchez, "Evaluating convolutional neural networks reliability depending on their data representation," in *Proc. 23rd Euromicro Conf. Digit. Syst. Design (DSD)*, Aug. 2020, pp. 672–679.

[108] Z. Wachter, "A cost effective motion platform for performance testing of MEMS-based attitude and heading reference systems," *J. Intell. Robot. Syst.*, vol. 70, nos. 1–4, pp. 411–419, Apr. 2013.

[109] C. A. Ippolito, S. Hening, S. Sankararaman, and V. Stepanyan, "A modeling, simulation and control framework for small unmanned multicopter platforms in urban environments," in *Proc. AIAA Model. Simulation Technol. Conf.*, Jan. 2018, p. 1915.

[110] R. Nallamalli, D. S. Chauhan, and C. Bhatnagar, "An approach to specify and test the control algorithm in focus framework," in *Proc. IEEE Conf. Open Syst. (ICOS)*, Nov. 2018, pp. 92–97.

[111] B. Dion and N. Dalmasso, "An integrated simulation platform to validate safety of autonomous vtol vehicles," in *Proc. Vertical Flight Soc.*, 2015, pp. 272–279.

[112] S. Ramakrishna, A. Dubey, M. P. Burruss, C. Hartsell, N. Mahadevan, S. Nannapaneni, A. Laszka, and G. Karsai, "Augmenting learning components for safety in resource constrained autonomous robots," in *Proc. IEEE 22nd Int. Symp. Real-Time Distrib. Comput. (ISORC)*, May 2019, pp. 108–117.

[113] Z. Xu, W. Li, and Y. Wang, "Robust learning control for shipborne manipulator with fuzzy neural network," *Frontiers Neurorobotics*, vol. 13, p. 11, Apr. 2019.

[114] B. Potteiger, H. Abdel-Aziz, H. Neema, and X. Koutsoukos, "Simulation based evaluation of security and resilience in railway infrastructure," in *Proc. 6th Annu. Symp. Hot Topics Sci. Secur.*, 2019, pp. 1–2.

[115] D. Meltz and H. Guterman, "Functional safety verification for autonomous UGVs—Methodology presentation and implementation on a full-scale system," *IEEE Trans. Intell. Vehicles*, vol. 4, no. 3, pp. 472–485, Sep. 2019.

[116] M. M. Sallami, M. I. Khedher, A. Trabelsi, S. Kerboua-Benlarbi, and D. Bettebghor, "Safety and robustness of deep neural networks object recognition under generic attacks," *Commun. Comput. Inf. Sci.*, vol. 1142, pp. 274–286, Dec. 2019.

[117] M. Ono, B. Rothrock, K. Otsu, and S. Higa, "MAARS: Machine learning-based analytics for automated rover systems," in *Proc. IEEE Aerosp. Conf.*, Mar. 2020, pp. 1–17.

[118] D. Cofer, I. Amundson, R. Sattigeri, A. Passi, C. Boggs, E. Smith, L. Gilham, T. Byun, and S. Rayadurgam, *Run-Time Assurance for Learning-Enabled Systems* (Lecture Notes in Computer Science), vol. 12229. Cham, Switzerland: Springer, 2020, pp. 361–368.

[119] E. Karlsson and N. Mohammadiha, "A data-driven generative model for GPS sensors for autonomous driving," in *Proc. 1st Int. Workshop Softw. Eng. Auto. Syst.*, May 2018, pp. 1–5.

[120] R. Schroeter and M. A. Gerber, "A low-cost VR-based automated driving simulator for rapid automotive UI prototyping," in *Proc. 10th Int. Conf. Automot. User Interface Interact. Veh. Appl.*, Sep. 2018, pp. 248–251.

[121] A. Bosio, P. Bernardi, A. Ruospo, and E. Sanchez, "A reliability analysis of a deep neural network," in *Proc. IEEE Latin Amer. Test Symp. (LATS)*, Mar. 2019, pp. 1–6.

[122] M. Elgharbawy, B. Bernier, M. Frey, and F. Gauterin, "An agile verification framework for traffic sign classification algorithms in heavy vehicles," in *Proc. IEEE/ACS 13th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Nov. 2016, pp. 1–8.

[123] D. R. Cacciagrano, F. Corradini, R. Culmone, N. Gorogiannis, L. Mostarda, F. Raimondi, and C. Vannucchi, "Analysis and verification of ECA rules in intelligent environments," *J. Ambient Intell. Smart Environ.*, vol. 10, no. 3, pp. 261–273, Jun. 2018.

[124] M. Karaduman and H. Eren, "Classification of road curves and corresponding driving profile via smartphone trip data," in *Proc. Int. Artif. Intell. Data Process. Symp. (IDAP)*, Sep. 2017, pp. 1–7.

[125] S. Noh and W.-Y. Han, "Collision avoidance in on-road environment for autonomous driving," in *Proc. 14th Int. Conf. Control, Autom. Syst. (ICCAS)*, Oct. 2014, pp. 884–889.

[126] A. Domina and V. Tihanyi, "Comparison of path following controllers for autonomous vehicles," in *Proc. IEEE 17th World Symp. Appl. Mach. Intell. Informat. (SAMI)*, Jan. 2019, pp. 147–152.

[127] A. Deshpande, K. Mathur, and E. Hall, "Comparison of three control methods for an autonomous vehicle," *Proc. SPIE*, vol. 7539, Jan. 2010, Art. no. 75390T.

[128] W. Qu, J. Xu, Y. Ge, X. Sun, and K. Zhang, "Development and validation of a questionnaire to assess public receptivity toward autonomous vehicles and its relation with the traffic safety climate in China," *Accident Anal. Prevention*, vol. 128, pp. 78–86, Jul. 2019.

[129] D. Hong, S. Kimmel, R. Boehling, N. Camoriano, W. Cardwell, G. Jannaman, A. Purcell, D. Ross, and E. Russel, "Development of a semi-autonomous vehicle operable by the visually-impaired," in *Proc. IEEE Int. Conf. Multisensor Fusion Integr. Intell. Syst.*, Aug. 2008, pp. 455–467.

[130] M. Bapat, M. Marathe, N. Khadloya, U. S. Karle, M. Karle, and M. R. Saraf, "Development of autonomous vehicle controller," SAE Tech. Papers 2019-26-0098, Jan. 2019.

[131] A. Farag, A. Hussein, O. M. Shehata, F. Garcia, H. H. Tadjine, and E. Matthes, "Dynamics platooning model and protocols for self-driving vehicles," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2019, pp. 1974–1980.

[132] H. Gao, H. Yu, G. Xie, H. Ma, Y. Xu, and D. Li, "Hardware and software architecture of intelligent vehicles and road verification in typical traffic scenarios," *IET Intell. Transp. Syst.*, vol. 13, no. 6, pp. 960–966, Jun. 2019.

[133] M. Seo and R. Lysecky, "Hierarchical non-intrusive in-situ requirements monitoring for embedded systems," in *Runtime Verification* (Lecture Notes in Computer Science), vol. 10548. Cham, Switzerland: Springer, 2017, pp. 259–276.

[134] R. Sun, S. Hu, H. Zhao, M. Moze, F. Aioun, and F. Guillemard, "Human-like highway trajectory modeling based on inverse reinforcement learning," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 1482–1489.

[135] S. Lee, H. Seo, H. Kwon, and H. Yoon, "Hybrid approach of parallel implementation on CPU–GPU for high-speed ECDSA verification," *J. Supercomput.*, vol. 75, no. 8, pp. 4329–4349, Aug. 2019.

[136] R. Bostelman and W. Shackleford, "Improved performance of an automated guided vehicle by using a smart diagnostics tool," in *Proc. IEEE Int. Conf. Ind. Technol.*, Mar. 2010, pp. 1688–1693.

[137] M. Z. M. Nasir, K. Hudha, M. Z. Amir, and F. A. A. Kadir, "Modelling, simulation and validation of 9 DOF vehicles model for automatic steering system," *Appl. Mech. Mater.*, vol. 165, pp. 192–196, Apr. 2012.

[138] M. Nishi, "Preemptive detection of unsafe motion liable for hazard," in *Proc. AAAI Workshops*, Jan. 2017, pp. 153–160.

[139] E. Cioroaica, T. Kuhn, and T. Bauer, "Prototyping automotive smart ecosystems," in *Proc. 48th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. Workshops (DSN-W)*, Jun. 2018, pp. 255–262.

[140] C. Wittpahl, H. B. Zakour, M. Lehmann, and A. Braun, "Realistic image degradation with measured PSF," *Electron. Imag.*, vol. 2018, no. 17, pp. 149-1–149-6, Jan. 2018.

[141] A. Joshi, "Real-time implementation and validation for automated path following lateral control using hardware-in-the-loop (HIL) simulation," SAE Tech. Paper 2017-01-1683, Mar. 2017.

[142] W. Wang, M. Cheng, and Y. Su, "Real-time stop sign detection and distance estimation using a single camera," in *Proc. 10th Int. Conf. Mach. Vis. (ICMV)*, Apr. 2018, Art. no. 106962A.

[143] D. M. Vavriv, O. O. Bezvesilniy, V. A. Volkov, A. A. Kravtsov, and E. V. Bulakh, "Recent advances in millimeter-wave radars," in *Proc. Int. Conf. Antenna Theory Techn. (ICATT)*, Apr. 2015, pp. 1–6.

[144] M. Acosta, S. Kanarachos, and M. E. Fitzpatrick, "Robust virtual sensing for vehicle agile manoeuvring: A tyre-model-less approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 3, pp. 1894–1908, Mar. 2018.

[145] P. Sharma, H. Liu, H. Wang, and S. Zhang, "Securing wireless communications of connected vehicles with artificial intelligence," in *Proc. IEEE Int. Symp. Technol. Homeland Secur. (HST)*, Apr. 2017, pp. 1–7.

[146] N. Silvis-Cividjian, "Teaching Internet of Things (IoT) literacy: A systems engineering approach," in *Proc. IEEE/ACM 41st Int. Conf. Softw. Eng., Softw. Eng. Edu. Training (ICSE-SEET)*, May 2019, pp. 50–61.

[147] T. Fehlmann and E. Kranich, "Theoretical aspects of consumer metrics for safety & privacy," *Commun. Comput. Inf. Sci.*, vol. 896, pp. 640–653, Dec. 2018.

[148] T. Abdelrahman and S. Courellis, "UAV formation maintenance in linear and curvilinear trajectories," in *Proc. AUVSI Unmanned Syst. North Amer. Conf.*, Las Vegas, NV, USA, 2012, pp. 1320–1323.
[149] M. Klās and A. Vollmer, "Uncertainty in machine learning applications: A practice-driven classification of uncertainty," in *Proc. Int. Conf. Comput. Saf., Rel., Secur.*, 2018, vol. 11094, pp. 431–438.
[150] Z. Wei, C. Wang, P. Hao, and M. J. Barth, "Vision-based lane-changing behavior detection using deep residual neural network," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 3108–3113.
[151] D. Ansari and G. Kochar, "Simulation of steering a self-driving car using 1) PID controller 2) neural network," in *Proc. Int. Conf. Smart Syst. Inventive Technol. (ICSSIT)*, Nov. 2019, pp. 212–218.
[152] H. L. S. Araujo, N. X. Verdezoto, S. Wali, C. D. N. Damasceno, R. Dimitrova, G. Kefalidou, M. Mehtarizadeh, M. R. Mousavi, J. Onime, J. O. Ringert, and J. M. Rojas, "Trusted autonomous vehicles: An interactive exhibit," in *Proc. IEEE Int. Conferences Ubiquitous Comput. Commun. (IUCC) Data Sci. Comput. Intell. (DSCI) Smart Comput., Netw. Services (SmartCNS)*, Oct. 2019, pp. 386–393.
[153] H. Guissouma, S. Leiner, and E. Sax, "Towards design and verification of evolving cyber physical systems using contract-based methodology," in *Proc. Int. Symp. Syst. Eng. (ISSE)*, Oct. 2019, pp. 1–8.
[154] K. Lim, T. Drage, C. Zhang, C. Brogle, W. Lai, T. Kelliher, M. Adina-Zada, and T. Braunl, "Evolution of a reliable and extensible high-level control system for an autonomous car," *IEEE Trans. Intell. Veh.*, vol. 4, no. 3, pp. 396–405, Dec. 2019.
[155] A. Harris, J. Stovall, and M. Sartipi, "MLK smart corridor: An urban testbed for smart city applications," in *Proc. IEEE Int. Conf. Big Data*, Dec. 2019, pp. 3506–3511.
[156] M. Zhu, Y. Wang, Z. Pu, J. Hu, X. Wang, and R. Ke, "Safe, efficient, and comfortable velocity control based on reinforcement learning for autonomous driving," *Transp. Res. C, Emerg. Technol.*, vol. 117, Aug. 2020, Art. no. 102662.
[157] K. Lim, T. Islam, H. Kim, and J. Joung, "A Sybil attack detection scheme based on ADAS sensors for vehicular networks," in *Proc. IEEE 17th Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2020, pp. 1–5.
[158] Y. C. Hou, K. S. M. Sahari, L. Y. Weng, H. K. Foo, N. A. A. Rahman, N. A. Atikah, and R. Z. Homod, "Development of collision avoidance system for multiple autonomous mobile robots," *Int. J. Adv. Robotic Syst.*, vol. 17, no. 4, Jul. 2020, Art. no. 172988142092396.
[159] J. Cao, C. Song, S. Peng, S. Song, X. Zhang, and F. Xiao, "Trajectory tracking control algorithm for autonomous vehicle considering cornering characteristics," *IEEE Access*, vol. 8, pp. 59470–59484, 2020.
[160] A. Sligar, "Machine learning-based radar perception for autonomous vehicles using full physics simulation," *IEEE Access*, vol. 8, pp. 51470–51476, 2020.
[161] C. Day, L. McEachen, A. Khan, S. Sharma, and G. Masala, "Pedestrian recognition and obstacle avoidance for autonomous vehicles using raspberry pi," in *Proc. Adv. Intell. Syst. Comput.*, 2020, vol. 1038, pp. 51–69.
[162] V. Riccio, G. Jahangirova, A. Stocco, N. Humbatova, M. Weiss, and P. Tonella, "Testing machine learning based systems: A systematic mapping," *Empirical Softw. Eng.*, vol. 25, no. 6, pp. 5193–5254, Nov. 2020.
[163] M. Bundin, A. Martynov, and F. Rumyantsev, "Legal framework for self-driving cars: The case of russia," in *Proc. 13th Int. Conf. Theory Pract. Electron. Governance*, Sep. 2020, pp. 206–213.
[164] X. Wang, J. Liu, T. Qiu, C. Mu, C. Chen, and P. Zhou, "A real-time collision prediction mechanism with deep learning for intelligent transportation system," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 9497–9508, Sep. 2020.
[165] R. B. Abdessalem, A. Panichella, S. Nejati, L. C. Briand, and T. Stifter, "Automated repair of feature interaction failures in automated driving systems," in *Proc. 29th ACM SIGSOFT Int. Symp. Softw. Test. Anal.*, Jul. 2020, pp. 88–100.
[166] D. De Dominicis and D. Accardo, "Software and sensor issues for autonomous systems based on machine learning solutions," in *Proc. IEEE 7th Int. Workshop Metrol. Aerosp.*, Jun. 2020, pp. 545–549.
[167] F. Rohde, "Continuous integration as mandatory puzzle piece for the success of autonomous vehicles," SAE Tech. Papers 2020-01-0087, Apr. 2020.
[168] N. Ahmad, A. Meng, and M. Sultan, "Applications of hardware-in-the-loop simulation in automotive embedded systems," SAE Tech. Paper 2020-01-1289, Apr. 2020.
[169] S. Bachuwar, A. Bulsara, H. Dossaji, A. Gopinath, C. Paredis, S. Pilla, and Y. Jia, "Integration of autonomous vehicle frameworks for software-in-the-loop testing," SAE Tech. Papers 2020-01-0709, Apr. 2020.
[170] M. Alcon, H. Tabani, J. Abella, L. Kosmidis, and F. J. Cazorla, "En-route: On enabling resource usage testing for autonomous driving frameworks," in *Proc. 35th Annu. ACM Symp. Appl. Comput.*, Mar. 2020, pp. 1953–1962.
[171] B. Vedder, J. Vinter, and M. Jonsson, "A low-cost model vehicle testbed with accurate positioning for autonomous driving," *J. Robot.*, vol. 2018, pp. 1–10, Nov. 2018.
[172] K. Liu, J. Gong, A. Kurt, H. Chen, and U. Ozguner, "A model predictive-based approach for longitudinal control in autonomous driving with lateral interruptions," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 359–364.
[173] J.-H. Ko, C.-S. Jeong, Y.-M. Jeong, and S.-Y. Yang, "A study on safety steering for 3 wheel autonomous vehicle," in *Proc. 12th Int. Conf. Control, Automat. Syst.*, Oct. 2012, pp. 708–711.
[174] A. Skarbek-Zabkin and M. Szczepanek, "Autonomous vehicles and their impact on road infrastructure and user safety," in *Proc. Int. Sci.-Tech. Conf. Autom. Saf.*, Apr. 2018, pp. 1–4.
[175] J. Li, J. Zhang, and N. Kaloudi, "Could we issue driving licenses to autonomous vehicles?" in *Proc. Int. Conf. Comput. Saf., Rel., Secur.*, 2018, vol. 11094, pp. 473–480.
[176] D. L. Luu, C. Lupu, and D. Chirita, "Design and development of smart cars model for autonomous vehicles in a platooning," in *Proc. 15th Int. Conf. Eng. Mod. Electr. Syst. (EMES)*, Jun. 2019, pp. 21–24.
[177] A. D'Amato, C. Pianese, I. Arsie, S. Armeni, W. Nesci, and A. Peciarolo, "Development and on-board testing of an ADAS-based methodology to enhance cruise control features towards CO_2 reduction," in *Proc. 5th IEEE Int. Conf. Models Technol. Intell. Transp. Syst. (MT-ITS)*, Jun. 2017, pp. 503–508.
[178] A. Papadoulis, M. Quddus, and M. Imprialou, "Evaluating the safety impact of connected and autonomous vehicles on motorways," *Accident Anal. Prevention*, vol. 124, pp. 12–22, Dec. 2019.
[179] K.-L. Lu, Y.-Y. Chen, and L.-R. Huang, "FMEDA-based fault injection and data analysis in compliance with ISO-26262," in *Proc. 48th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. Workshops (DSN-W)*, Jun. 2018, pp. 275–278.
[180] L. Murray, "Hitch a ride with Harry," *Eng. Technol.*, vol. 12, no. 7, pp. 62–65, 2017.
[181] F. Bock, S. Siegl, and R. German, "Mathematical test effort estimation for dependability assessment of sensor-based driver assistance systems," in *Proc. 42th Euromicro Conf. Softw. Eng. Adv. Appl. (SEAA)*, Aug. 2016, pp. 222–226.
[182] Y. Gao and T. Gordon, "Optimal control of vehicle dynamics for the prevention of road departure on curved roads," *IEEE Trans. Veh. Technol.*, vol. 68, no. 10, pp. 9370–9384, Oct. 2019.
[183] A. Jafri and J.-W. Kim, "Proving the capability of arm IP for functional safety applications," in *Proc. 19th Int. Workshop Microprocessor SOC Test Verification (MTV)*, Dec. 2018, pp. 1–5.
[184] C. Wilcox and B. Williams, "Runtime verification of stochastic, faulty systems," in *Proc. Int. Conf. Runtime Verification*, vol. 6418, 2010, pp. 452–459.
[185] P. Koopman and B. Osyk, "Safety argument considerations for public road testing of autonomous vehicles," SAE Tech. Papers 2019-01-0123, Apr. 2019.
[186] J. E. R. Condia and M. S. Reorda, "Testing permanent faults in pipeline registers of GPGPUs: A multi-kernel approach," in *Proc. IEEE 25th Int. Symp. On-Line Test. Robust Syst. Design (IOLTS)*, Jul. 2019, pp. 97–102.
[187] S. Narla, "The evolution of connected vehicle technology: From smart drivers to smart cars to. Self-driving cars," *J. Inst. Transp. Eng.*, vol. 83, no. 7, pp. 22–26, 2013.
[188] F. Ahmed-Zaid, H. Krishnan, M. Maile, L. Caminiti, S. Bai, J. Stinnett, S. VanSickle, and D. Cunningham, "Vehicle safety communications–applications: Multiple on-board equipment testing," *SAE Int. J. Passenger Cars Mech. Syst.*, vol. 4, no. 1, pp. 547–561, Apr. 2011.
[189] F. Ahmed-Zaid, H. Krishnan, M. Maile, L. Caminiti, S. Bai, J. Stinnett, S. Vansickle, and D. Cunningham, "Vehicle safety communications-applications: Multiple on-board equipment testing," SAE Tech. Papers2011-01-0586, Apr. 2011.
[190] A. Widner and G. Bári, "Investigating the effects of roll centre height, during transient vehicle maneuvers," in *Proc. 16th Mini Conf. Vehicle Syst. Dyn., Identificat. Anomalies*, Budapest, Hungary, 2019, pp. 255–260.

[191] X. Zhao, V. Robu, D. Flynn, K. Salako, and L. Strigini, "Assessing the safety and reliability of autonomous vehicles from road testing," in *Proc. IEEE 30th Int. Symp. Softw. Rel. Eng. (ISSRE)*, Oct. 2019, pp. 13–23.
[192] A. M. Nascimento, A. C. M. Queiroz, L. F. Vismari, J. N. Bailenson, P. S. Cugnasca, J. B. Camargo Junior, and J. R. de Almeida, "The role of virtual reality in autonomous Vehicles' safety," in *Proc. IEEE Int. Conf. Artif. Intell. Virtual Reality (AIVR)*, Dec. 2019, pp. 50–57.
[193] A. M. Boggs, R. Arvin, and A. J. Khattak, "Exploring the who, what, when, where, and why of automated vehicle disengagements," *Accident Anal. Prevention*, vol. 136, Mar. 2020, Art. no. 105406.
[194] E. Cheek, H. Alghodhaifi, C. Adam, R. Andres, and S. Lakshmanan, "Dedicated short range communications used as fail-safe in autonomous navigation," *Proc. SPIE Unmanned Syst. Technol.*, vol. 11425, Apr. 2020, Art. no. 114250P.
[195] R. Raveendran, K. B. Devika, and S. C. Subramanian, "Intelligent fault diagnosis of air brake system in heavy commercial road vehicles," in *Proc. Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2020, pp. 93–98.
[196] X. Zhao, K. Salako, L. Strigini, V. Robu, and D. Flynn, "Assessing safety-critical systems from operational testing: A study on autonomous vehicles," *Inf. Softw. Technol.*, vol. 128, Dec. 2020, Art. no. 106393.
[197] F. Utesch, A. Brandies, P. Pekezou Fouopi, and C. Schießl, "Towards behaviour based testing to understand the black box of autonomous cars," *Eur. Transp. Res. Rev.*, vol. 12, no. 1, p. 1, Dec. 2020.
[198] N. Xama, M. Andraud, J. Gomez, B. Esen, W. Dobbelaere, R. Vanhooren, A. Coyette, and G. Gielen, "Machine learning-based defect coverage boosting of analog circuits under measurement variations," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 25, no. 5, pp. 1–27, Oct. 2020.
[199] R. Kannan and R. Lasky, "Autonomous vehicles still decades away: 2019," in *Proc. Microelectron. Symp. (Pan Pacific)*, Feb. 2020, pp. 1–6.
[200] I. Kurzidem, A. Saad, and P. Schleiss, "A systematic approach to analyzing perception architectures in autonomous vehicles," in *Proc. Int. Symp. Model-Based Saf. Assessment* (Lecture Notes in Computer Science), vol. 12297. Cham, Switzerland: Springer, 2020, pp. 149–162.
[201] P. Koopman and M. Wagner, "Toward a framework for highly automated vehicle safety validation," SAE Tech. Papers 2018-01-1071, Apr. 2018.
[202] P. Koopman and M. Wagner, "Autonomous vehicle safety: An interdisciplinary challenge," *IEEE Intell. Transp. Syst. Mag.*, vol. 9, no. 1, pp. 90–96, Spring 2017.
[203] F. Flammini, S. Marrone, R. Nardone, M. Caporuscio, and M. D'Angelo, "Safety integrity through self-adaptation for multi-sensor event detection: Methodology and case-study," *Future Gener. Comput. Syst.*, vol. 112, pp. 965–981, Nov. 2020.
[204] J. Zhang and J. Li, "Testing and verification of neural-network-based safety-critical control software: A systematic literature review," *Inf. Softw. Technol.*, vol. 123, Jul. 2020, Art. no. 106296.
[205] M. Alawadhi, J. Almazrouie, M. Kamil, and K. A. Khalil, "A systematic literature review of the factors influencing the adoption of autonomous driving," *Int. J. Syst. Assurance Eng. Manage.*, vol. 11, no. 6, pp. 1065–1082, Dec. 2020.

NIJAT RAJABLI was born in Gakh, Azerbaijan, in 1997. He received the bachelor's degree in computer engineering from Khazar University, Baku, Azerbaijan, in 2018. He is currently pursuing the master's degree in software technology with Linnaeus University, Växjö, Sweden. From 2016 to 2017, he has attended the Middle East Technical University, Ankara, Turkey, as a Selected Exchange Student of computer engineering. His experience with software engineering and AI in industry includes an internship at Ericsson in 2020 and his current work at Softwerk, where he is researching applications of AI on timber log tracing. His research interests include AI and verification of deep learning systems.

FRANCESCO FLAMMINI (Senior Member, IEEE) was born in Formia, Italy, in 1978. He received the master's degree *(cum laude)* in computer engineering and the Ph.D. degree in computer and automation engineering from the University of Naples Federico II, Italy, in 2003 and 2006, respectively. From 2003 to 2016, he was a Software V&V Engineer from 2003 to 2007 and a Senior Innovation Engineer from 2007 to 2016 with Ansaldo STS (now Hitachi Rail). From 2016 to 2017, he worked with the Italian State Mint and Polygraphic Institute as an Information Security Compliance Manager. Since 2018, he has been working as an Associate Professor with Linnaeus University, Sweden, where he has chaired the Cyber-Physical Systems (CPS) environment. Since 2020, he has also been a Professor of computer science with Mälardalen University, Sweden. He had leadership roles in more than ten research projects, has edited or authored more than ten books and 130 publications. His research interests include resilient CPS and trustworthy autonomous systems. He is also an ACM Distinguished Speaker, a member of several IEEE societies, including Intelligent Transportation Systems.

ROBERTO NARDONE was born in Napoli, Italy, in 1986. He received the master's degree in computer engineering and the Ph.D. degree in computer and automation engineering from the University of Naples Federico II, Italy, in 2009 and 2013, respectively. Since 2018, he has been an Assistant Professor with the Mediterranean University of Reggio Calabria, Italy. He currently serves as an Associate Editor of distinguished journals, such as IEEE Access and *Journal of Universal Computer Science*, and has coauthored more than 60 research articles published in peer-reviewed international journals and conference proceedings. His research interest includes quantitative evaluation of non-functional properties by means of model-driven techniques.

VALERIA VITTORINI was born in Napoli, Italy, in 1965. She received the master's degree in mathematics and the Ph.D. degree in computer and automation engineering from the University of Napoli Federico II, Italy, in 1991 and 1995, respectively. Since 2005, she has been an Associate Professor with the University of Napoli Federico II. She has coedited proceeding volumes and coauthored more than 80 research papers published in international journals and conference proceedings. Her research interests include model-driven engineering, formal modeling, and verification of critical systems. She is recently investigating the application of artificial intelligence to the transportation sector, and in particular to railway systems. She currently serves as the Coordinator for the H2020 Shift2Rail project RAILS.