Early Reliability Assessment of AI-based Automotive Systems

Early Reliability Assessment of AI-based Automotive Systems

SHAILESH SUDHAKARA HEGDE, DINESH CYRIL SELVARAJ, JOSIE E. RODRIGUEZ CONDIA, NICOLA AMATI, CARLA FABIANA CHIASSERINI, FRANCESCO DEFLORIO, and MATTEO SONZA REORDA, Politecnico di Torino - Center for Automotive Research and Sustainable Mobility, Turin, Italy

The availability of powerful Artificial Intelligence (AI) algorithms boosts the development of advanced functionalities in the automotive domain and is essential to enable the deployment of autonomous and semi-autonomous decision-making vehicles. However, integrating such advanced and complex functionalities in automotive systems is challenging due to several factors, including: *i)* the mandatory compliance with strict safety regulations, which require effective strategies to ensure timely development while allowing thorough dependability evaluations, and *ii)* the short time-to-market imposing limited development, verification, and validation periods. In particular, the analysis of the effects of faults affecting the hardware executing an AI-based application is made challenging by the target system's complexity (in terms of both hardware and software). Reliability analysis often resorts to Fault Injection techniques. However, Fault Injection experiments are often unacceptably time-consuming and limited to some components of the overall system, thus failing to consider the fault impact at the vehicle level. This work proposes a new method, named *Two-steps IntegrAted Reliability Assessment* (*TIARA*), for early estimation of the impact at the vehicle level of faults affecting the hardware running AI-based perception tasks in the automotive domain. TIARA allows for the early exploration and evaluation of algorithms, driving agents, and critical operational scenarios. TIARA can estimate the effects of faults affecting a subsystem up to the vehicle level, integrating a fault injection approach at the neural network level with a commercial automotive-grade virtual scenario generator. When compared to previous works, TIARA dramatically reduces the required computational effort by adopting a two-stage evaluation strategy. It first performs static analysis to determine fault vulnerabilities and identify the most vulnerable parts (code blocks) in a targeted application. Then, it focuses on the most susceptible parts of the neural network and estimates system-level effects on vehicle dynamics by combining the system's perception, control, and driving features. We validated our methodology through the exhaustive evaluation of two applications: *Lane Centering Assistance* (LCA) and *Emergency Lane Keeping Assistance* (ELKA), using the *YoloP* model for perception. The experimental results on nine different driving scenarios show that TIARA allows for an effective early estimation of systems reliability through relevant driving dynamics and comfort metrics, as mandated by standards, while reducing computing complexity by up to 43.2X in comparison with a fully-exhaustive evaluation approach. In addition, the validation of the TIARA methodology through a hardware-in-the-loop implementation shows that the results closely match the behavior of a real-world system, demonstrating the versatility of the TIARA strategy for the evaluation of automotive systems.

CCS Concepts: • **Computer systems organization → Reliability**; • **Hardware → Board- and system-level test**.

Additional Key Words and Phrases: Automotive, Artificial intelligence, Reliability evaluation, Lane Keeping assistance, Lane Centering Assistance, Soft errors

---

A preliminary version of this work has been presented to IEEE LATS 2025 [27].
Authors' Contact Information: Shailesh Sudhakara Hegde; Dinesh Cyril Selvaraj; Josie E. Rodriguez Condia; Nicola Amati; Carla Fabiana Chiasserini; Francesco Deflorio; Matteo Sonza Reorda, Politecnico di Torino - Center for Automotive Research and Sustainable Mobility, Turin, Italy , name.surname@polito.it.

Manuscript submitted to ACM

ACM Reference Format:
Shailesh Sudhakara Hegde, Dinesh Cyril Selvaraj, Josie E. Rodriguez Condia, Nicola Amati, Carla Fabiana Chiasserini, Francesco Deflorio, and Matteo Sonza Reorda. 2025. Early Reliability Assessment of AI-based Automotive Systems. 1, 1 (July 2025), 34 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction and Motivations

The ability to perform a fast and effective evaluation of functional safety and reliability is a primary concern when developing new *Advanced Driver Assistance Systems* (ADAS), which represent a vital domain where the Internet of Things (IoT) paradigm is applied. This concern is becoming even more critical as vehicles get equipped with increasingly sophisticated electronic devices supporting software-defined paradigms to continuously enhance driving capabilities, user comfort, and system autonomy [36, 39, 58]. Furthermore, the wide adoption of *Artificial Intelligence* (AI) greatly boosts the system's capabilities. On the other side, AI systems rise additional challenges in such safety-critical scenarios, where reliability is a major concern.

In particular, some AI algorithms (e.g., *Convolutional Neural Networks* or CNNs) are now widely employed to enable complex tasks, such as object recognition and sensor fusion, which are pivotal to the automatic control and system management in ADAS [7, 10]. Due to their operational complexity, data-intensive operations, and high computational power, AI applications usually exploit advanced hardware acceleration platforms with extended capabilities, including embedded FPGAs, Vector Processors in CPUs, *Neural Processing Units* (NPUs), *Tensor Processing Units* (TPUs), *Graphics Processing Units* (GPUs) or customized chips (e.g., Tesla's *Full Self-Driving* or 'FSD') [11, 16, 17, 43, 57]. The devices used to support AI algorithms are often based on advanced semiconductor technologies (e.g., manufactured with 7nm processes, or less), which are known to be more susceptible to hardware faults than those previously used in automotive [3, 24, 25, 30, 48, 56]. This makes the need for effective reliability evaluation techniques even more important than for other IoT systems. Unfortunately, both the data-intensive nature of the operations and the demand for high computational power in AI applications increase the challenges in assessing their reliability and guaranteeing the correct system operation during the in-field operation. Furthermore, AI in the automotive domain must satisfy industry dependability requirements (i.e., ISO 26262 and ISO/IEC 22989 [22, 47]) to handle the different threats affecting the system dependability. In this scenario, a key role is played by the clever adoption of strategies to evaluate the reliability features, adapting them to the specific characteristics of AI-powered applications [50]. In addition, the stages of development, design exploration, and early reliability assessment for automotive systems with respect to faults in the hardware are costly and require the interaction between several engineering departments, leading to long development times, from unit design to system verification [17]. This situation exacerbates the need for frameworks to support the early evaluation of design and reliability parameters in AI systems (e.g., sensor fusion and driving agents exposed to errors), which can also be used to support possible design improvements.

1.1 State of the art and existing gaps

Several works have addressed the need for frameworks and strategies to speed up the development, verification, and validation of applications in automotive. In particular, most industrial-grade and academic frameworks for automotive focus on verifying the functional properties in the different vehicle and support design stages. In contrast, few frameworks include the support to evaluate non-functional properties, such as assessing the impact on reliability stemming from faults affecting the hardware. The available frameworks (often characterized by a high degree of automation) can be

grouped into three categories: *i)* hardware emulation systems, *ii)* hybrid (software and hardware) systems, and *iii)* software-based simulators.

The hardware-based frameworks (based on hardware emulation or *Hardware-in-the-Loop* (HIL)) combine real components such as engines, actuators, sensors, and board computers with early-stage design solutions to verify and validate the real-time operational features of autonomous and semi-autonomous systems. These frameworks usually exploit specialized testbeds and instrumentation facilities for real-time (in-the-loop) system evaluation [6]. Moreover, adopting configurable platforms, such as FPGAs, is also desirable for quickly evaluating early-stage designs and systems. In this context, [44] describes an emulator using FPGAs for the early codesign and functional verification of software and hardware components for automotive, which, however, neglects reliability assessment. In fact, the evaluation of reliability at these levels is commonly limited, mainly due to the very few fault models that can be considered in an autonomous and semi-autonomous system at this stage (e.g., interconnect failures or system perturbations) without compromising the costs and the integrity of the adopted testbed.

The second category (*hybrid*) combines the performance of HIL solutions with the flexibility of software frameworks to analyze the operation of autonomous and semi-autonomous systems. By simulating environmental elements, such as vehicular traffic scenarios, these ecosystems enable the pseudo-real-/real-time evaluation and verification of system functional properties [9, 34, 60]. Unfortunately, these solutions require costly and elaborated testbed facilities (e.g., complete vehicles or some parts of them, working in controlled environments) with sophisticated interconnected systems, specialized instrumentation, equipment, and custom frameworks, thus typically exhibiting high complexity [1, 6] with limited capabilities to analyze system failures due to component faults. In addition, the high cost of the testbed components and the system's complexity reduce the possibility of performing dedicated analyses and evaluations of non-functional parameters (e.g., reliability and dependability) without compromising the system's integrity.

The third category of frameworks (*Software simulation*) are the most flexible and allow easy integration of several automotive components, such as controller agents [18], in-vehicle networks, and internal and external sensing features, while still ensuring reasonable fidelity of the operational environment. Specifically, these frameworks provide mechanisms to evaluate, analyze, and simulate the interaction of the different automotive system components. Moreover, simulation environments can interact with each other, as well as with users (e.g., through augmented reality [55]), and are flexible enough to update and deploy system configurations with minimal effort [8, 28]. Similarly, the minimal costs involved in developing and deploying components and features on simulator-based platforms enable the easy adoption of these frameworks for reliability assessment purposes with acceptable levels of accuracy. A simulator for the functional verification and validation of algorithms and software solutions for connected autonomous and semi-autonomous systems has been introduced in [53]. This framework, however, focuses only on assessing the operational features of a system while the non-functional properties, such as dependability or reliability, are neglected. In contrast, other works have proposed strategies and tools to address the resiliency characterization in automotive systems [29]. For instance, [34] has proposed a framework to evaluate the resiliency, safety, error propagation, and masking properties of autonomous and semi-autonomous systems. The tool supports the usage of some fault models on fundamental components of a vehicle under road/traffic scenarios defined by sensor inputs, object perception tasks, sensor fusion, planner, controller, and machine learning algorithms. Similarly, [35] presents a fault injection framework (*AVFI*) for the holistic evaluation of traffic violations due to faults/errors in autonomous and semi-autonomous components like LiDAR/camera sensors, communicating paths, actuators, and AI-based perception, localization, and motion planning. The framework combines an autonomous driving simulator (*CARLA*) [19] with an AI-based autonomous Agent. However, the tools used for generating the operational scenario, such as *CARLA*, *IsaacSim* [45], or *Open Pilot* [12], often

oversimplify the dynamics of autonomous and semi-autonomous systems. Moreover, the limited sensor support of these tools leads to an inaccurate representation of the behavior of modern autonomous and semi-autonomous systems. Additionally, the above mentioned tools can barely model faults affecting the complex computational hardware used in the most recent AI-based automotive applications. Finally, these simulators often require considerable computational power, which increases exponentially when considering reliability assessment objectives.

In conclusion, the following technical and scientific gaps still exist. *First*, most existing frameworks and strategies for evaluating and validating autonomous and semi-autonomous systems focus on their functional system properties while neglecting other features, including reliability. *Second*, the few existing strategies focusing on evaluating the resilience of AI-powered software and hardware blocks require complex frameworks. Since such frameworks typically require to consider a huge number of possible faults, they demand an exceedingly high computational effort that makes the evaluation of AI applications (e.g., perception) reliability in realistic automotive traffic/road scenarios hardly feasible . *Third*, existing works have overlooked crucial aspects, namely, the impact of hardware faults on autonomous and semi-autonomous systems and their evaluation at the system level via the adoption of relevant automotive indicators and metrics, such as users' vehicle accelerations, acceleration angles, and ride comfort.

More in general, given the complexity of the addressed system, it is extremely challenging to estimate the impact of faults, whose effects may propagate through different application levels, in a reasonable time and with acceptable accuracy. An automotive system is indeed composed of sensors (e.g., cameras, radars, LiDARs) whose data are processed in real-time by very sophisticated hardware (e.g., GPUs), executing different types of applications (e.g., neural networks and control algorithms). Additionally, such applications should enable autonomous driving of vehicles that have their own dynamic characteristics. Consequently, current strategies seriously suffer from the huge computational effort required to evaluate all possible fault sources in a such a complex system. To reduce the overall execution complexity, some of the existing approaches resort to sampling strategies. However, a straightforward fault sampling method may lead to a simplified assessment of the target AI-based system reliability, and it may be insufficient to identify the most vulnerable system components.

1.2 Our approach

To address the existing gaps, this work proposes a new strategy for the early-stage design exploration and reliability estimation of AI-based applications for automotive systems. Specifically, our methodology, named *Two-steps IntegrAted Reliability Assessment* (*TIARA*), identifies the most critical error sources on the target AI-based application and effectively determines their impacts at the vehicle system level.

The TIARA strategy uses two steps (*Static analysis* and *Dynamic evaluation*) to provide affordable evaluation times while preserving characterization accuracy. The first step focuses on characterizing and identifying the most vulnerable structures in the targeted AI-based application inside an autonomous or semi-autonomous system (e.g., the most sensitive layers in a Neural Network). To this end, it might adopt exhaustive, pseudo-exhaustive, statistical, or custom fault injection campaign approaches [14, 23, 46, 52] to identify the most vulnerable parts of an AI-based application. This early estimation of the most susceptible parts allows the identification of those stages in the AI software that are the most relevant for the application to provide a correct output. In addition, this step dramatically reduces the number of faults that have to be considered for the dynamic evaluation, while preserving high accuracy. The second step is, instead, a dynamic analysis conducted through fault injection campaigns focusing only on the most vulnerable parts (e.g., the Neural Network layers identified in step 1) and characterizing the corruptions induced by faults in the hardware composing the closed-loop system and implementing the vehicle dynamics. Thanks to this approach, the

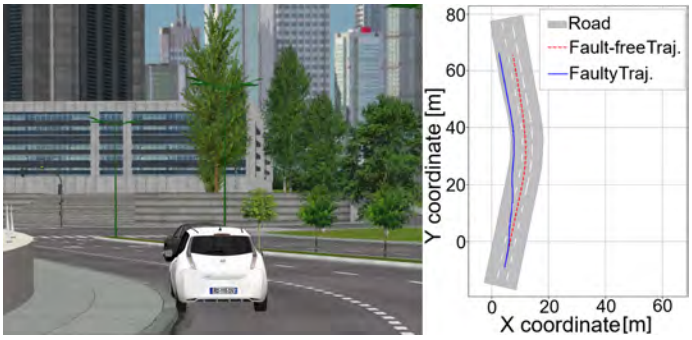


Fig. 1. Left: Example of a driving scenario we analyze through our framework, with a car traveling on a winding road. Right: the resulting car trajectory in the case with no faults and in the case where a fault occurs.

TIARA methodology effectively trades off accuracy with the computational effort to analyze large AI-based automotive applications.

To experimentally evaluate and validate our proposed approach, we developed a custom and flexible framework by integrating an automotive-grade traffic/road scenario simulator with fault injection capabilities to represent hardware errors in the system, sensor models, vehicle dynamics, and driving agents in the loop. We then evaluated the reliability of the whole closed-loop system, considering two representative AI-powered applications: *Lane Centering Assistance* (or LCA) , and *Emergency Lane Keeping Assistance* (ELKA) . The LCA aims to keep a vehicle centered in a road lane by resorting to the data coming from the vehicle's camera [61]. Instead, ELKA traces the critical closeness of the vehicle to lane lines and road borders using camera sensors, and it provides corrective steering adjustments to avoid unintentional lane departures.

To show the flexibility of our approach, we evaluated the target system (including the *YoloP* neural network for both LCA and ELKA) under three representative driving scenarios, namely, *straight*, *winding*, and *with obstacle*, each under three different lightening/weather condition cases, namely, *daylight*, *night*, and *rainy*.

Our results from the static analysis indicate that the most vulnerable layers in the AI software (e.g., faults that can significantly impact the system behavior) are part of specific structures in YoloP, specifically, *Backbone*, *Head*, and a few layers of the *Neck*). Other structures are instead less susceptible to system corruption, namely, most layers in the *Neck* and other perception heads. Then, the results obtained through our dynamic evaluation demonstrate that road/traffic features can substantially impact the overall system reliability when faults arise in perception, and they may lead to high-risk situations due to relevant lateral deviations of the vehicle. It is worth noting that thanks to the TIARA's first step, we have substantially reduced the number of faults to be considered in the latter dynamic evaluation of the system. As an example, Figure 1 depicts the corruption in the vehicle's navigation trajectory due to a fault impacting the perception (*YoloP*) for a driving scenario with curves (winding).

Then, we evaluated the computational complexity of the TIARA strategy against an exhaustive strategy, where all possible faults were considered. The validation indicates that TIARA reduced by about 43.2 times the required time to exhaustively perform a reliability assessment on the system. In addition, we compared the effectiveness of TIARA in the closed-loop system, showing its superior ability in identifying the most critical faults with respect to a statistical fault injection strategy. Finally, we validated TIARA through a HIL implementation, demonstrating its ability to correctly classify fault effects via simulation experiments. Upon acceptance of the work, a database reporting the reference (fault-free) experiments and the corruption effects for each scenario will be available in a public repository.

1.3 Our contributions and paper organization

The main contributions of this work can be summarized as follows:

- We describe the TIARA methodology, which allows an early reliability estimation of the impact of hardware faults on the vehicle-level behavior produced by AI-based applications in automotive systems. TIARA comprises two steps (*Static* and *Dynamic*). The *Static* step identifies the AI algorithm's most susceptible parts (layers). Then, the *Dynamic* step evaluates the closed-loop automotive system by focusing only on the faults affecting the parts selected by the first step, thus significantly reducing the required computational effort and time. Therefore, TIARA can work with reasonable time requirements without compromising the accuracy of results. An exhaustive evaluation across nine road scenarios (targeting all parameters in the perception application) showed that TIARA reduces the computational complexity by up to 43.2× compared to an exhaustive fault-injection strategy, while accurately assessing fault impacts in the closed-loop system.
- We introduce a framework tool implementing the TIARA strategy for the early reliability evaluation of AI-based applications in automotive systems. Our flexible framework combines two environments to allow the static characterization of AI-based applications and then perform the closed-loop dynamic evaluation of an automotive system by integrating one automotive-grade scenario generator with fault injection capabilities, sensor models, one perception task, and a control agent to drive the vehicle.
- By using TIARA, we characterize the reliability of two AI-based perception applications: *Lane Centering Assistance* (LCA) and *Emergency Lane Keeping Assistance* (ELKA). The experimental results, based on over $7.14 \cdot 10^6$ (bit-flip) faults injected into the neural network weights across nine road scenarios, show that a small fraction of faults, ranging approximately from 0.2% to 0.9%, in highly sensitive regions of the AI-based perception model (specifically, the *Backbone* and *Head* layers of the YoloP network) can critically compromise the overall system reliability for both applications.
- We perform the early reliability assessment of the target AI-powered automotive systems, using different dynamics indicators in automotive (*lane offset*, *steering wheel angle*, and *lateral acceleration*) and one complementary comfort metric (*equivalent acceleration*). This effectively enables the early identification of the most critical faults and to study their effects on AI-based applications for perception. Furthermore, our approach allows the ranking of different scenarios according to their criticality. Finally, it is worth remarking that our analysis allows further design and validation steps to focus on the most critical scenarios only, reducing the overall system development time.
- We validate our proposed TIARA strategy through a Hardware-in-the-Loop (HIL) platform that includes the automotive perception system. The experimental results of the HIL evaluation show equivalent fault vulnerabilities to those identified by simulation, demonstrating TIARA's effectiveness for early-stage evaluation, exploration, and characterization of real-world automotive applications.

An early version of the TIARA strategy has appeared in the conference paper [27]. Besides presenting a refined methodology with respect to [27], we now account for other representative performance indicators related to vehicle dynamics and passengers' comfort, and we benchmark the obtained results against performance thresholds established by standards [31]. Moreover, we demonstrate the generality of our approach by considering an additional perception-based application, namely, ELKA, and substantially extend our performance evaluation. While doing so, we analyse corner-case automotive road scenarios and rank the considered faults according to the criticality of their effects in terms of automotive indicators and comfort metrics. We also derive the computational complexity of TIARA and compare it

to that of an exhaustive fault injection strategy; we benchmark TIARA's effectiveness against that of a purely statistical strategy. Finally, in this work, we validate TIARA's strategy and its applicability to system-level reliability assessment using a HIL implementation, further demonstrating its suitability for automotive early-stage evaluation.

The rest of the manuscript is organized as follows. Sec. 2 provides some background regarding the primary challenges for the reliability assessment in modern systems, including automotive. Moreover, Sec. 2 briefly describes the fundamentals of AI-based perception tasks and the *YoloP* architecture. Then, Sec. 3 introduces the proposed TIARA methodology, while Sec. 4 details the implemented framework. Sec. 5 describes the experimental setup, and Sec. 6 presents and discusses experimental results. Sec. 7 illustrates and benchmarks TIARA's computational cost and effectiveness . Sec. 8 validates the TIARA strategy through a HIL implementation, while Sec. 9 provides a final discussion of the proposed approach. Finally, Sec. 10 draws some conclusions and highlights future research directions.

2 Challenges and Background

This section first discusses challenges for reliability estimation in modern systems. Then, it describes automotive threats regarding functional safety and fault impact from the hardware, software, and regulations points of view. Finally, it provides an overview of the YOLOP neural network [62] for automotive perception, which, given its relevance, we take as use case.

2.1 Challenges in the Characterization and Estimation of Reliability in AI-based Automotive Systems

Advances in semiconductor technology scaling drive the design and integration of energy-efficient, high-performance systems that are essential for the deployment of AI-based IoT applications, including those in safety-critical domains like the automotive one. However, next-generation devices (e.g., those manufactured with technology nodes below 7 nm) are highly vulnerable to physical defects and faults induced by temporal and environmental variations during in-field operation. Such faults may be due to different causes, such as premature aging, unexpected wear-out, or high sensitivity to external radiation, which may substantially impact the system's resilience [3, 24, 25, 30, 48, 56].

A key step towards solving such system vulnerabilities consists in performing reliability characterization on the systems. In fact, reliability assessments and characterizations are common industrial practices that support the evaluation and analysis of the effects of possible faults impacting the system. The primary objective of reliability characterization is to identify the most vulnerable hardware structures and software code blocks – those that, if affected by a fault, could critically compromise in-field system performance. The results of these analyses are then used to identify and implement fault countermeasures and hardening mechanisms that can improve the overall system resilience.

Figure 2 illustrates how a physical *defect* inside a vehicle's component can propagate throughout the system and cause software errors and system faults. Specifically, a physical defect in basic components, such as a transistor, can disrupt hardware operations and lead to a *fault* in the local system modules (e.g., logic gates or memory cells), making them deviate from their expected behavior. The interaction between faulty hardware and various software layers can propagate the defect's effects and cause *errors* at the software level, which causes observable deviations in software variables from their expected values. These errors may further propagate and corrupt the behavior of other components and sub-systems, potentially resulting in systematic *failures* that are critical deviations from intended system functionality and can be particularly critical in safety-sensitive applications, including those in automotive [4]. In some cases, the intrinsic features of the hardware, the software code, or system-level redundancies may mask the impact of faults. However, the growing complexity of new devices, characterized by higher transistor densities, increases both the probability of occurrence of faults and the difficulty of accurately estimating their probability. Additionally,

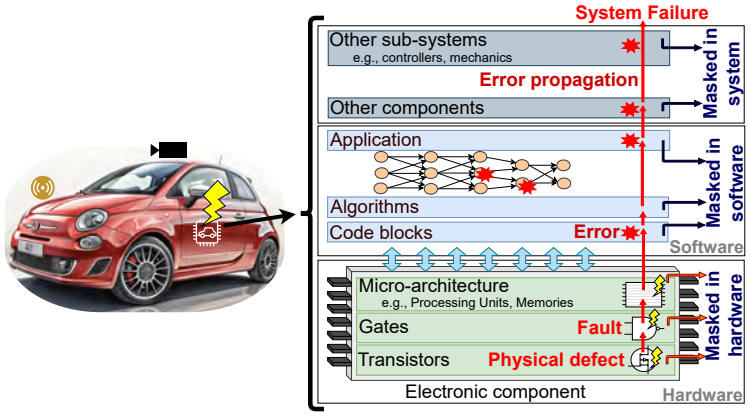


Fig. 2. Graphical representation of the propagation of a physical defect inside a component through an AI-based automotive system and its effects in terms of software errors and system failures.

adopting complex technologies like AI, which rely on large-scale data processing and data-intensive computations, complicates the analysis of fault propagation effects, making reliability characterization increasingly challenging. This paper specifically addresses this issue by analyzing fault propagation mechanisms in complex AI-based systems. In our work, we, therefore, focus on the latter task and propose an effective and efficient method for analyzing fault propagation impacts in an AI-based system.

Existing strategies for assessing and estimating system reliability leverage a range of approaches, from formal methods to experimental-based techniques. Formal evaluation relies on theoretical foundations and mathematical or logical models to describe and predict how faults can impact a system. They are particularly effective whenever an accurate mathematical or physical model of the system is available. Experimental-based methods, on the other hand, assess fault effects by empirically evaluating representative system models or the system itself [51]. While effective for early estimation, these methods require multiple independent experiments (i.e., extensive fault injection campaigns) on each targeted system component (e.g., hardware unit, code block, or software application). These campaigns use specialized frameworks to automate the experiments, but simulation times increase linearly with the complexity of the targeted component and the number of faults to evaluate, thus seriously suffering from the huge computational effort required to consider all possible faults in a system. Consequently, traditional experimental reliability characterization becomes impractical for AI-based applications, where modern devices contain a very large number of transistors, and AI applications demand intensive data processing. For instance, previous studies have shown that performing reliability characterization of a small Neural Network running on a GPU would require an unacceptable amount of computational time – exceeding 10,000 days [15]. This highlights and exacerbates the need for more efficient solutions to assess reliability in complex AI-based applications and systems, such as those used in the automotive domain.

2.2 Safety and Reliability Threats in Automotive Systems

As illustrated in Figure 3(a), AI-based automotive applications, such as AI-based control agents and perception tasks, require the extensive use of multi-sensor platforms (e.g., exploiting radars, LiDARs, cameras, ultrasonic), which may generate and process massive amounts of data but may be vulnerable to faults occurrence. As discussed above, a fault may originate at any time and propagate its effect to the software, other components, and sub-systems, to finally

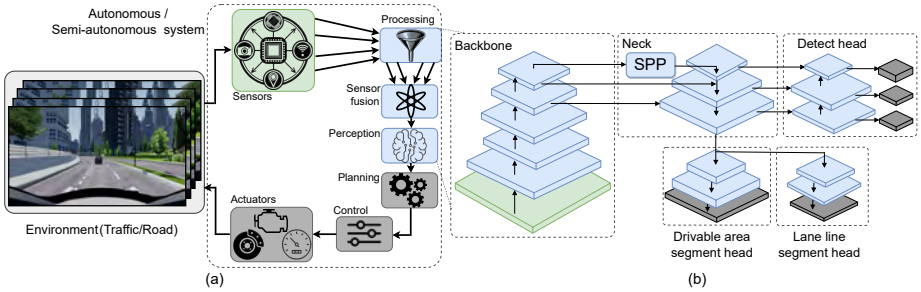


Fig. 3. An example of Autonomous/Semi-autonomous systems (a); YOLOP architecture (b).

disrupt the system behavior during, e.g., sensor fusion and perception tasks. Similarly, faults might propagate across the Planning and Control agents, reaching the actuators and impacting the overall system operation, including the vehicle's Lateral and Longitudinal accelerations, Steering wheel angle, and Lane offset detection.

The combination of these factors, namely, the data-intensive nature of the application, the increasing complexity of the system, and the fault sensitivity of the technology, makes the reliability evaluation of an automotive system a particularly challenging task. This is explicitly addressed by several strict safety and reliability automotive standards like ISO 26262. Additionally, the impact of faults in the vehicle's behavior varies significantly depending on the system configuration, AI algorithm, fault location (e.g., in the perception or the control agent), and the fault type (e.g., whether it is permanent/transient). As a consequence, the susceptibility of automotive systems to hardware faults affecting their underlying components (e.g., CPUs and hardware accelerators, including GPUs) while running an AI-based application is still far from being fully understood [34].

The previous observations motivate the exploration of efficient techniques for the reliability analysis and evaluation of complex automotive systems during early design phases. The main goal there is the identification of the most vulnerable structures and code blocks, thus guiding the development of effective countermeasures.

2.3 Automotive Perception through YOLOP

Among AI-based applications, computer vision tasks emerge as particularly relevant. Such AI-powered algorithms enable vehicles to perceive the surrounding environment by processing in real-time multi-modal data collected through a number of sensors [38].

An example of an AI model for computer vision tasks is the YOLOP *(You Only Look Once for Panoptic driving perception)* model [62], which we also use in our work. YOLOP can simultaneously perform object detection, drivable area segmentation, and lane line detection for automotive driving. Its architecture comprises three main sections: *1)* **Backbone**, *2)* **Neck**, and *3)* **Head**, as depicted in Figure 3(b).

The **Backbone** includes 14 layers and extracts multi-scale features from input images using a deep Convolutional Neural Network based on the YOLO architecture [49]. This stage can be optimized for handling diverse perception tasks concurrently. The **Neck** stage comprises 19 layers. It focuses on the refinement and enrichment of features across different layers, enhancing the ability to detect varying-size objects through *Spatial Pyramid Pooling* (SPP) layers [26]. Finally, the **Head** section uses independent heads per task: one for object detection inside the driving scene, such as vehicles, pedestrians, and traffic signs (*Detection*), one for drivable areas on the road (*Drivable Area Segmentation*), and one for lane boundaries (*Lane Line Detection*). The efficient organization of the YOLOP architecture enables the reuse of

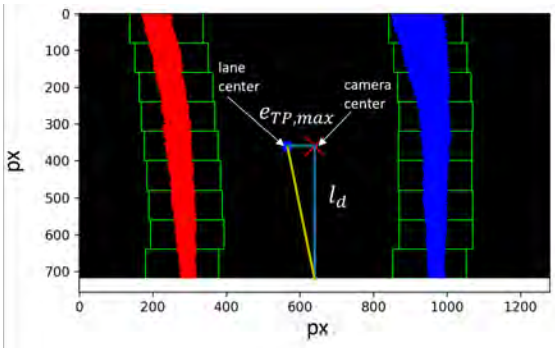


Fig. 4. An example of drivable lane center offset calculation by processing the lane line detection frames from YoloP.

some layers, specifically those in the **Backbone** and **Neck**, to process several features while using independent heads for specific tasks.

In our work, we focus on applications involving perception. For this purpose, the heads in YOLOP model can be customized to identify crucial driving information for a vehicle, such as the drivable lane center. As depicted in Figure 4, this is performed by transforming the pixel coordinates into physical coordinates, i.e., by employing a gain (G_{px2m}) parameter that represents the meters per pixel in **x** dimension for look-ahead distance (l_d).

To compute such a dynamic performance indicator, a vehicle equipped with one or more camera sensors mounted at some specific locations (e.g., near the center rear-view mirror) first identifies in each frame captured by the camera the region of interest for lane detection. This is the area in which lane boundaries are expected to appear. Such an area of interest is commonly marked using a trapezoid-shaped polygon for isolation. Then a perspective transformation procedure transforms the original area of interest into a bird's-eye view image. Moreover, a sliding window image processing strategy accumulates the pixels of the required lane lines. A pixel's histogram is computed in the area of interest to determine those histogram values that are different from zero. In fact, the peak values within the pixel's histogram of an image identify the initial coordinates of the left and right windows. Then a sliding window is applied to collect and identify all the lane line pixels with a defined window width and height. The coordinates of the next sliding window are determined based on the mean of the pixel points in the previous sliding window.

For the computation of the drivable lane center, high levels of accuracy are desirable, which are obtained by selecting a proper number of windows (e.g., 8). Then, a second-order polynomial fitting and transformation process identifies the lane lines, and subsequently, the lane center line is calculated.

3 The TIARA Strategy

Our strategy, named TIARA, aims to evaluate the behavioral impacts of faults affecting the hardware of an Autonomous or Semi-autonomous system. As illustrated in Figure 5, the two phases are *(i)* ***Static Analysis*** and *(ii)* ***Dynamic Evaluation***. The *Static Analysis* aims at identifying the most vulnerable structures in the AI-powered application, i.e., the layers inside a multi-class ML application actively executing a targeted task, such as perception or control. This phase consists of a set of fault injection experiments targeting the main elements (i.e., the weights in a layer) of each section of the ML model. To speed up these experiments, the analysis is performed using static images, instead of the continuous flow of video frames coming from the vehicle cameras. Further, the experiments focus on exhaustive error evaluation to identify the most susceptible code blocks in the AI application under different operational scenarios. It is worth noting that this step might be adapted to use statistical or custom approaches of fault injection and evaluation.

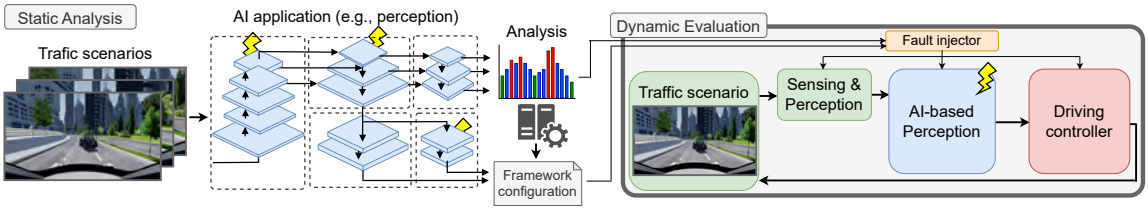


Fig. 5. Schematic representation of the proposed TIARA strategy.

The *Dynamic Evaluation* phase, instead, addresses *only* those code blocks of the AI-based application that, during the static analysis, have been detected as most critical and vulnerable to faults. In this phase, our method aims to evaluate the fault effects at the system level. To this end, we adopt a closed-loop approach, modeling all system components: driving scenario, sensors, perception agent, and control agent. It follows that the experiments for the dynamic evaluation are significantly more time-consuming than those performed for the static analysis. Our two-phase approach is, therefore, key to ensuring a feasible reliability characterization of the system, as the static analysis allows us to focus on and assess only the impact of relevant faults, which, as shown later, represent a small fraction of all possible fault instances that may occur. Moreover, our strategy allows us to efficiently perform early estimation of vulnerabilities during system development under various scenarios to identify the most suitable design improvements.

In the following, we detail the TIARA strategy.

3.1 Static Analysis

This phase evaluates the impact of faults only on the AI-based application without considering the role and effect of the control agent on board the vehicle. We inject faults in the AI application by placing errors (i.e., bit flips) in the AI model weights, output feature maps, or a combination of both. In [46], we demonstrated that a proper spatial and temporal distribution of these errors can model the effects of faults affecting the underlying hardware. Thus, TIARA draws on the results of previous works [23, 59] to include support for injecting both permanent and transient faults in the underlying hardware. In more detail, the approach uses a limited set of highly representative input stimuli, such as video frames from crucial driving situations. After each fault simulation, the environment collects the outputs of the AI-based application to determine corruption effects, if any. Then, the errors that impact the application's output most significantly are correlated with the internal application layers to determine the most highly vulnerable parts of the application AI model, as depicted in Figure 5 (top).

Importantly, as further detailed in Section 5.2, the level of relevance of the errors is assessed according to their impact on the AI application output through the computation of representative performance indicators. Such indicators, also referred to hereinafter as key performance indicators (KPI), may include lane center offset or lateral acceleration, depending on the examined perception function. We compare the value of such KPIs in the case where a fault is injected against the case of fault-free execution of the AI-based application. The fault effect is then classified as *(i)* `Silent Data Corruption` (SDC), if the fault causes a variation of any of the KPIs, or as *(ii)* `Masked`, if the fault has no impact on the KPIs. Next, those layers of the AI application, in which faults lead to SDCs, are identified as most vulnerable and classified as *'highly sensitive'*. Fault injection instances affecting highly sensitive layers of the AI application are selected for the dynamic, closed-loop evaluation of the system, which is detailed in the next section.

3.2 Dynamic Evaluation

This second phase extends the analysis *from the AI application to the system level*. Specifically, the evaluation aims to determine the system disruptions caused by faults in the hardware that executes the AI application, identifying the effect of error propagation throughout the entire automotive system during driving operations. It is worth remarking that our focus is on corruptions leading to system misbehavior, originated from faults inside the system.

To this end, our system evaluation makes use of a vehicle simulator that includes AI applications for navigation or control purposes, as well as a control agent for autonomous driving maneuvers. An automotive-grade road generator builds scenarios that represent surrounding vehicles, varying traffic conditions, and sensing device interaction (e.g., cameras, LiDAR) on board the vehicle. The method is illustrated in Figure 5 (bottom), which underscores how the dynamic evaluation of the AI application is based on the closed-loop operation of the automotive system, which captures real-time environmental data, such as lane markings through camera sensors. The captured video frames are then processed by the AI application to detect and estimate features of interest, e.g., the lane center in an LCA application. Based on the collected features, the vehicle's control agent adjusts the vehicle's movements to navigate the road environment effectively.

For the evaluation, a preliminary fault-free simulation computes the baseline behavior of the system as well as the AI application's effectiveness in navigating a specific driving condition. Since our analysis focuses on the corruption effects in the system caused by faults, the fault-free operation of the system serves as a reference to distinguish such effects from other anomalies unrelated to faults, e.g., uncertainties or rare events. Then, several statistical fault injection campaigns [50] are performed to characterize the system under dynamic and changing driving conditions. These campaigns are executed by injecting errors into the AI application's highly sensitive layers identified during the statistical analysis, i.e., those causing SDCs.

Since the AI application's assessment aims to determine the impact of the errors and their propagation on the closed-loop system, we automated the evaluation process through a framework that integrates the scenario generator and automatically evaluates the system's KPIs. The framework allows for the system evaluation under several typical and critical maneuvers and traffic scenarios. For instance, in an LCA application, lateral acceleration, steering wheel angle, and lane deviation are the representative KPIs according to industrial standards [31, 32].

Finally, the degradation of the KPIs is analyzed to determine the critical effects causing failures of the automotive navigation system.

4 The TIARA Framework

This section describes the framework tool we developed to implement the proposed TIARA method for system reliability characterization. The framework comprises two environments, one for the static analysis and one for the dynamic evaluation, and uses *Python* to orchestrate the two phases.

The static analysis environment integrates a custom software-based fault injector based on *PytorchFI* [41]. The fault injector supports different fault models, namely, bit-flips in the weights and feature maps of the AI-based application [46, 50]. In detail, the fault model adopted in TIARA (which has been widely used in many previous works in the literature) represents hardware faults arising in the registers and memories (e.g., RAM) inside the system's ADAS, which are crucial structures storing weights and feature maps related to the AI-based perception application. The effects of such faults may propagate during the in-field operation of the AI-based perception, thus affecting the vehicle behavior, or be masked at different levels (e.g., by the neural network behavior, the control agent, or the vehicle dynamic).

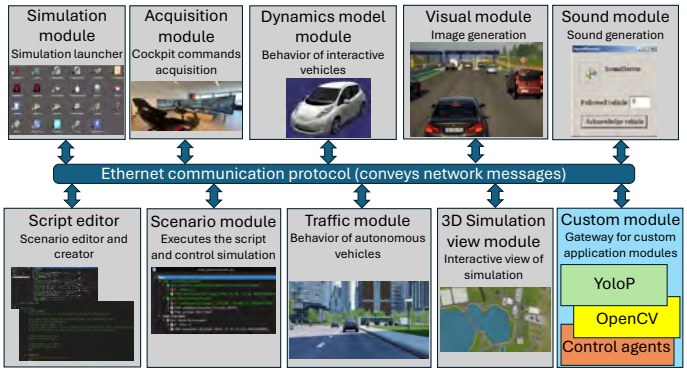


Fig. 6. Overview of the SCANeR architecture integrated into the TIARA framework.

Importantly, the environment provides a mechanism to target specific weights or layers of the application AI model, allowing for exhaustive or partial evaluation of all weights. For each assessment, the environment injects one fault, e.g., one or several bit-flips, through a fault mask before executing the AI application. The fault mask acting on weights consists of a scalar operation between a targeted weight from a layer and a corrupting mask, which is expressed as a fixed hexadecimal or integer value representing the error effect to be applied. Similarly, the fault mask acting on feature maps applies a corrupting mask on the map according to the hardware-aware error effects that need to be simulated [46]. The static analysis environment first injects a fault in the AI application and then evaluates its effects when processing some representative video frames from the simulated scenario. The framework is then restarted, the previous fault is removed, and another fault is considered. The implemented environment incorporates multi-threading and distributed computing schemes to speed up the process. It is worth noting that the framework can be further extended and adapted to support more elaborate or custom fault models to address faults arising from other components inside ADAS systems.

The second environment implements the dynamic evaluation and is based on an automotive-grade realistic virtual scenario generator (*SCANeR* by *AVSimulation* [5, 20]), high-fidelity sensor models, realistic vehicle dynamics, driving controller agents, and image processing components to assess closed-loop system reliability of the vehicle on typical driving/traffic scenarios with accurate environmental conditions. *SCANeR* is a comprehensive platform designed to simulate and evaluate the driving behavior commonly used to develop and verify ADAS systems and AD&SAD technologies, which also enables the study of the behavior of a complete vehicle moving in real-world driving scenarios.

Figure 6 depicts the SCANeR architecture, showcasing module interactions and the ability to include custom modules (shown at the bottom-right), which is crucial for the implementation of the TIARA strategy. In detail, the dynamic environment integrates custom modules implementing the AI-based application, some complementary frame processing tasks, the fault injector, and the control agents. The frame processing operations are performed through the *OpenCV* Computer Vision library. Moreover, the fault injector uses a similar injection scheme as in the static analysis step. In this case, the dynamic environment injects one fault in the AI application before starting the simulation of a given scenario. Then, the vehicle navigates through the scenario. By looking at the closed-loop system, TIARA enables the analysis of the effects of fault propagation and their impacts on vehicle dynamics. Once the vehicle navigates for the time interval specified by the target driving scenario (e.g., the time required to perform a curve trajectory), the dynamic environment records the execution time series, classifies the fault effects, and finally restarts the simulation, targeting a new fault. As mentioned, one golden vehicle's navigation (fault-free operation) in the scenario serves as a reference for the fault classification.

It is worth noting that during the experiments for the dynamic evaluation, only faults identified as relevant by the static analysis are considered. Identifying such faults is based on critical effects (i.e., impacts on performance indicators). In the first case, the critical effects directly target the faults, causing higher deviations on the evaluated vehicle's dynamics, e.g., those faults causing the most significant trajectory error offset.

5 Experimental Setup

This section describes the driving scenarios we considered for our experiments. Then, it describes the KPIs we used to assess the faults' impact on the vehicle behavior.

5.1 Targeted AI-based Applications and Road Scenarios

As case studies, we configured the framework to evaluate the impact of faults on two different applications: Lane Centering Assistance (LCA) and Emergency Lane Keeping Assistance (ELKA). The LCA application aims at keeping a vehicle centered with respect to the road lane edges it is occupying, which are collected through camera sensors. On the other hand, the ELKA application is an ADAS feature that helps keep a vehicle from departing its lane or road edge in emergency conditions, such as when a lane change or accident is imminent. In the experimental framework we developed, both applications use an AI-based perception subsystem based on YoloP. The YoloP perception network employs the lane line segment head to identify lines in the video frames captured by the vehicle traveling along the road. More in detail, we tuned up the YoloP application to process 20 frames/s, considering one camera sensor with 1,280×720 pixels resolution mounted on the front of the vehicle.

Furthermore, we configured *SCANeR* to evaluate nine scenarios, which are commonly proposed and employed by vehicle manufacturers and standards in the market [33, 42, 63] to validate vehicles' performance and safety features. The nine scenarios comprise three vehicle navigation cases using a 3-lane highway with intermittent line marking, namely, *straight* (S), *winding* (W), and *with obstacle* (O), each under three different light and weather conditions (*daylight*, *night*, and *rainy*). To evaluate the sensitivity to possible faults in the target system, each traffic scenario includes a target actor, i.e., the vehicle under control, and one external actor, i.e., another vehicle in the environment, moving at 12 km/h and 60 km/h, for LCA and ELKA, respectively.

The *straight* (S) scenario represents ideal driving conditions, since both the perception and the driving agent operate without external disturbances caused by the surrounding environment. Hence, this scenario aims to evaluate the direct impact of a fault on the perception task and its propagation, possibly corrupting the vehicle's driving performance in the absence of external obstacles and external critical features (e.g., a curve).

The *winding* (W) scenario forces the interaction of the vehicle and its driving agent with the surrounding environment. In this scenario, two factors are considered: *i)* the control agent operation and *ii)* the impact of a fault in the perception and its propagation effects on the vehicle's driving performance. For our evaluation, we deploy a simple but effective lateral controller agent to follow the road in the scenario. In particular, the road comprises a curvature to the left, forcing the vehicle and its driving control agent to act and follow the road. Thus, the scenario allows the evaluation of the impact of faults in the presence of an implicit control complexity stemming from acting on a winding road.

Finally, the scenario *with obstacle* (O) allows the evaluation of the interaction between a vehicle, the surrounding environment, one external vehicle interacting in the scene (and possibly masking the lane markings), and a fault affecting the AI-based perception application.

For the static analysis, representative frames are selected according to the most crucial circumstances per scenario. For example, the O scenario is evaluated considering critical frames showing the lane-crossing by the external vehicle.

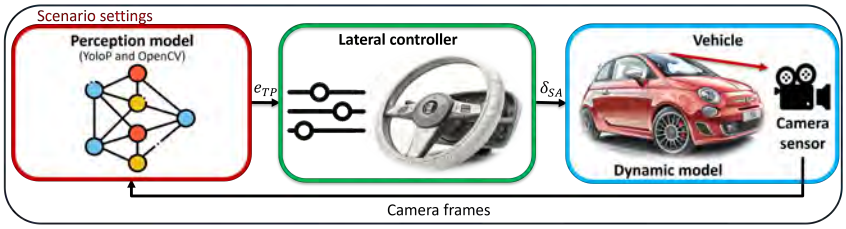


Fig. 7. The controller block diagram for lateral control in the dynamic evaluation. e_{TP} is the lane center line offset error at the target point and δ_{SA} is the steering wheel angle.

For all scenarios, the dynamic evaluation is performed using a simulation time of 23 s; unlike what happens in the static phase, the camera continuously feeds the perception model at a rate of 20 frames per second for each scenario. During the evaluation, the `S` and `W` scenarios place the targeted actor in an initial position, and each application keeps it in the lane. In contrast, the `O` scenario consists of a straight road with the external actor changing line (approaching from the right) after 2 s from the start of the evaluation. In order to create more realistic scenarios and challenge both applications, we configured the graphics engine with the *Unreal engine* in the simulation and scenario generator (*SCANeR*) after a preliminary analysis of the scene quality to provide enhanced details of the scenes. These include realistic weather conditions and mirror-reflection effects in the camera sensors.

In the experiments, we used a simple and effective non-linear feedback controller as a control agent for the vehicle's lateral control, as depicted in the scheme of Figure 7. The controller is given by:

$$\delta_{SA} = -\text{atan}\left(\frac{e_{TP}}{l_d}\right) \cdot \frac{\delta_{SA}^{\max}}{\delta_{sa}^{\max}} \tag{1}$$

where δ_{SA} is the steering wheel angle (control signal), e_{TP} is the lane center line offset error (control error) at the target point, l_d is the look-ahead distance, $\delta_{SA}^{\max}$ is the maximum steering wheel angle, and $\delta_{sa}^{\max}$ is the maximum steering angle.

5.2 Dynamic Automotive Indicators

We employed several KPIs associated with vehicle dynamics to quantify and rank the criticality of each fault in every scenario of the evaluated system. We remark that the KPIs we considered, besides being based on industrial standard documentation, provide complementary information. Some represent the impact of faults on the overall trajectory of the vehicle. Others, such as accelerations, are instead related to the vehicle's dynamics and can be calculated and observed only during the perception and control tasks. Additionally, the rich set of KPIs we consider allows us to evaluate not only safety aspects, but also the level of passenger comfort.

In particular, the TIARA framework has been instrumented to compute five main automotive metrics, which are briefly described as follows:

- **Lane Keeping:** it assesses the vehicle's ability to stay within its lane according to a predefined or reference navigation trajectory. A fault in the system may cause the vehicle trajectory to unduly move out of the expected lane.
- **Lateral Deviation:** it measures the distance between the vehicle's current path and the desired (detected center line) path. A fault in the system may cause offset effects in the detected center line.

- **Lateral Acceleration (LACC):** it represents the rate of change of the vehicle's velocity in the lateral direction. A fault in the vehicle might cause deviations from its practical operative ranges according to the type of vehicle and road conditions.
- **Steering Wheel Angle (SA):** it indicates the effort needed to maneuver the vehicle according to the desired trajectory [2, 32]. A fault in the system might cause offsets in the angle.
- **Equivalent Acceleration (a_{eq}):** it measures the comfort and smoothness of the ride, considering acceleration components (longitudinal and lateral) according to ISO2631 [31]. In detail, a_{eq} is the Euclidean norm of frequency-weighted root mean squared acceleration components [40]. A fault in the system might impact the acceleration, causing peaks during vehicle navigation.

It is worth noting that we associate a different set of KPIs for the evaluation of error impacts on each application. Specifically, the LCA's corruption effects are evaluated on critical variations on the lateral deviation, SA, LACC, and a_{eq} metrics, while the lane keeping is used to analyze the effects in ELKA.

6 Experimental Results

This section presents the experimental results obtained with the TIARA framework in the 9 scenarios introduced in the previous section, considering the different KPIs listed above.

During the static analysis, we exhaustively inject around $7.94 \cdot 10^6$ soft errors (bit-flips) targeting all the weights of the YoloP sections (*Backbone*, *Neck*, and *Head*). In detail, we use a standard approach based on flipping the Most Significative Bit (MSB) per weight, only [50]. As input to the experiments, the static analysis makes use of a restricted set of representative video frames from each targeted scenario. Then, we perform a dynamic evaluation to characterize the impact of errors on the vehicle, for both the LCA and ELKA applications . In the experiments, we inject 9,000 soft errors (1,000 per scenario), corresponding to the most critical faults, based on the results of the first phase. This sampling corresponds to an interval of confidence of 95% and 1% error margin [37]. The above figures are clearly pessimistic since the sampling is not random.

The experiments required an equivalent computational time of around $6.65 \cdot 10^3$ hours; specifically, 735.3 hours for the static and 3.7 hours for the dynamic evaluations for each scenario and weather condition, considering single-core execution. In practice, the computational cost of the static phase amounted to around 73 h per scenario . The computational tasks were distributed to a 6-node cluster with 2 Intel 16-core Xeon Scalable Processors Gold 6130 (2.10 GHz) equipped with 6 NVIDIA Tesla V100 SXM2 and 32 GB of RAM. The dynamic phase employed instead a workstation with an 8-core Intel i7-11700K (3.6 GHz) processor, 32 GB of RAM, and one NVIDIA Geforce $RTX3080_{Ti}$ GPU with 12 GB of RAM. A detailed analysis of the performance costs is reported in Section 7.1.

6.1 Static Analysis

The static analysis classifies the fault effects on the YoloP outputs by examining the magnitude of the e_{TP} parameter, i.e., the lane center line offset error at the target point for each one of the nine considered scenarios. Then the analysis identifies the most sensitive layers, according to the error magnitude. The results of such a static analysis are then used in the second step to evaluate the propagation of a subset of the faults and their effects on the automotive system for the LCA and ELKA applications.

Figure 8 illustrates the distribution of the maximum distance between the reference (golden) vehicle's center and the lane center line offset error due to faults impacting the weights of the AI-based perception task on all analyzed

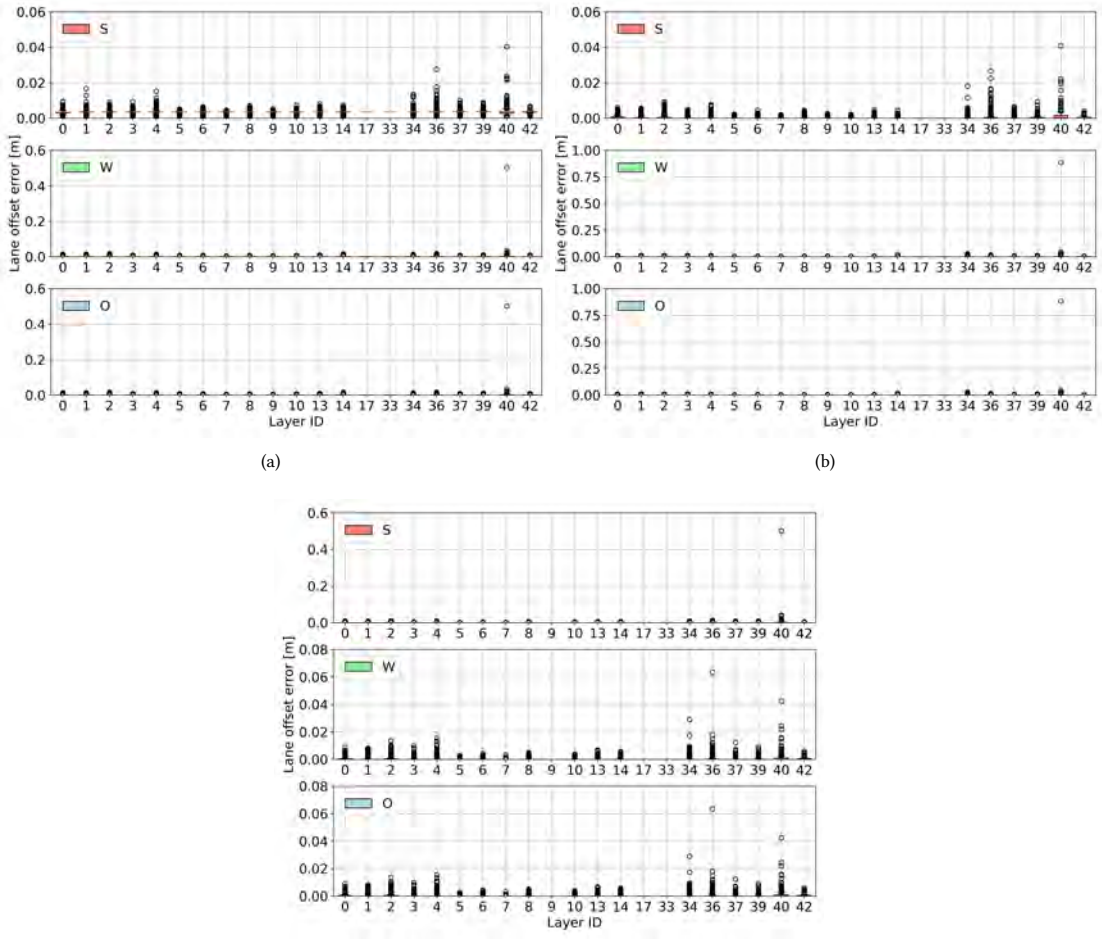


Fig. 8. Distribution of the lane center line offset error on the layers of YoloP for the straight (S), winding (W), and with obstacle (O) traffic scenarios for **daylight** (a), **night** (b), and **rainy** (c) weather conditions.

scenarios. On the horizontal axis, we report the different layers in the AI application according to their position in the system, starting from the layers near the input. The vertical axis depicts the variability and median per layer, providing insights into the distribution of lane center offsets. Layers from 15 to 33 are not shown in the plots because faults affecting them produced no offset corruption in any of the considered scenarios.

Our experimental results indicate that the most sensitive layers in YoloP, causing the most significant deviations from the lane center, are consistent across all scenarios and are located in the *Backbone* (layers 0–7), parts of the *Neck* (layers 8–14), and the *Head* (layers 34–42) sections. Furthermore, we identify some highly sensitive layers (namely, layers 6, 8, 34, 36, and 40), which are highly prone to causing large deviations for most of the scenarios.

In addition, during our experimental analysis, we observed for all analyzed road scenarios the intrinsic fault tolerance of some layers in the *Neck* of YoloP: layers 15 to 33 are not shown because faults did not produce any offset corruption. In other words, all faults affecting these layers were classified as masked. A detailed analysis of the YoloP structure,

focusing on this group of layers, shows that the inference process does not directly employ some of these layers to infer and identify road lines. Instead, some of these layers are primarily employed for complementary inference tasks associated with road-drivable areas and object detection in the scenario, thus explaining their resilience to errors.

Interestingly, the results indicate that the maximum error deviation directly depends on the characteristics of the analyzed scenario. In particular, the error magnitudes up to 0.041 m for S, 0.50 m for W, and up to 1.30 m for O. Moreover, a detailed analysis of the most critical error effects (large deviation magnitude) by evaluating the top 5% of the critical errors indicates that *Backbone* layers (near the input) are more sensitive in the W and O scenarios. In contrast, the *Head* layers are more prone to errors in the S scenario. The most sensitive layers are the primary injection targets during the dynamic evaluation.

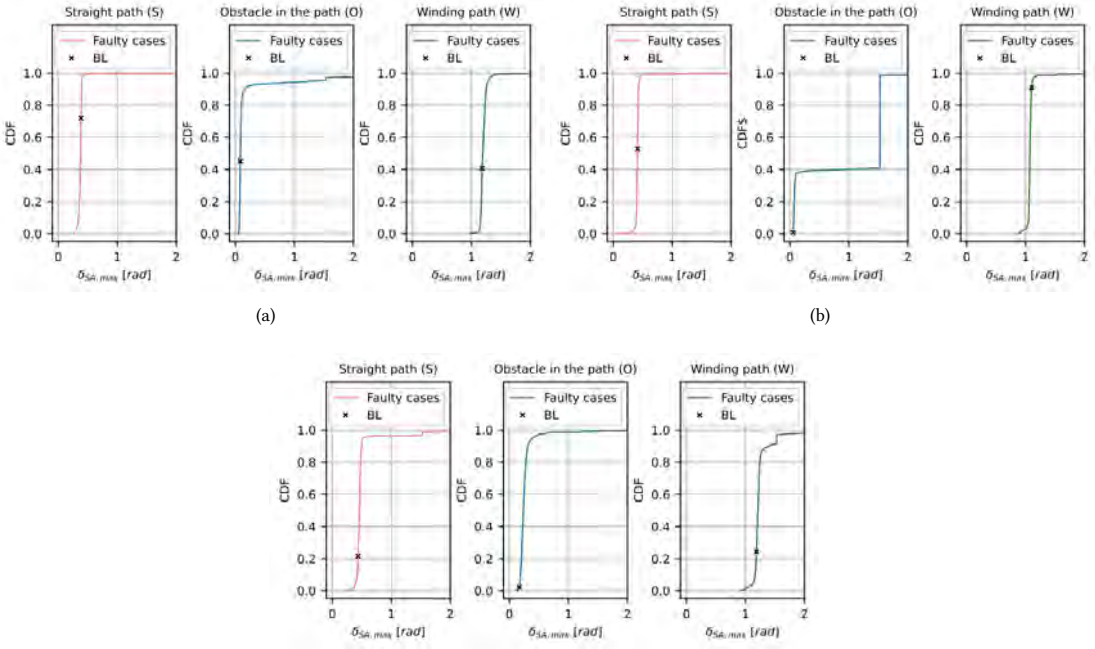
6.2 Dynamic Evaluation

This section analyses the propagation of fault effects from the perception towards the system for both applications, and their impact on the car's behavior by computing dynamic metrics associated with the vehicle's driving performance.

6.2.1 Lane Centering Assistance . In the closed-loop experiments, we compute three metrics associated with the vehicle's driving performance: *(i)* the *Steering wheel Angle* (SA), *(ii)* the *Lateral Acceleration* (LACC), and *(iii)* the *Equivalent Acceleration* (a_{eq}). Since our focus is on the analysis of the critical effects on the system due to faults in the perception tasks, which are associated with large magnitude deviations in the dynamic metrics, we first calculate the *Cumulative Distribution Function* (CDF) of the maximum effects for each indicator by considering the time series from the 1,000 dynamic evaluation experiments. The CDF allows us to observe the maximum (critical) corruption in the vehicle's dynamics due to a faulty perception and its possible impacts on the vehicle's navigation trajectory. In general, the results highlight a different behavior of the system when the perception structure is faulty, depending on the driving scenario.

Figure 9 illustrates the CDF of maximum SAs for all dynamic experiments (Faulty cases) , for each scenario. The plots also illustrate the performance of our baseline (BL), i.e., the fault-free operational reference of the system . Overall, the CDF results indicate that the maximum SA deviation from the baseline strongly depends on the evaluated scenario and its weather conditions. Considering that more CDF cases near zero, on the left of BL, represent lower critical impacts , one can observe that the **daylight** condition yields mostly stable performance in all driving scenarios. Indeed, a moderate percentage of the considered faults, from around 41% to 71%, cause lower deviations (less than 0.3 rad) compared to the baseline SA, and the faults causing a significant deviation in SA (e.g., higher than one rad.) are less than 10% of all considered ones. Conversely, **night** driving scenarios show a contrasting susceptibility to faults with considerable variation concerning the baseline SA from about 10%, in W scenario, to 99% in WO scenario. Interestingly, faulty SA cases in WO, higher than 1.5 rad, are observed for around 60% of the dynamic evaluations, indicating a higher criticality of faults in some driving scenarios and weather conditions. Regarding **rainy** driving scenarios, they yield slight SA deviations due to faults. In detail, the SA corruption varies from around 75% to 96% of the evaluated faults. Interestingly, in the **rainy** driving scenarios, a limited percentage of faults (from around 5% to 15%) causes a SA variation higher than 0.3 rad.

The previous results indicate that the combination of scenario and weather condition plays a crucial role in the system's impacts on SA when faults affect the perception. To corroborate these impacts, we computed the CDF of the maximum LACC for each dynamic scenario (Figure 10). The impacts on LACC are moderate in **daylight** and **rainy** driving scenarios, with, respectively, about 63% to 75% and 45% to 70% of faults causing no, or just slight (less than



(a)

(b)

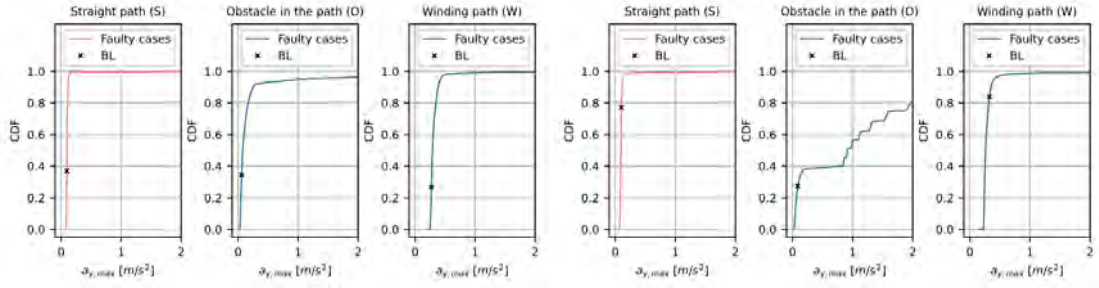
(c)

Fig. 9. CDF of the maximum steering wheel angle ($\delta_{SW,max}$) for the daylight (a), night (b), and rainy (c) weather conditions on each evaluated dynamic scenario (S, O, W). Fault-free baseline (BL) values for daylight conditions in S, O, and W scenarios are 0.39, 0.08, and 1.18 rad, respectively. Baseline values for night conditions in S, O, and W scenarios are 0.41, 0.05, and 1.1 rad, respectively. Baseline values for rainy conditions in S, O, and W scenarios are 0.44, 0.17, and 1.19 rad, respectively.

0.2 m/s^2), variations in the LACC maximum magnitude. In detail, less than 10% of faults cause large LACC variations that might affect the vehicle trajectory in **daylight** scenarios. In contrast, significant effects of LACC on **rainy** driving scenarios directly depend on the navigation features of each scenario. Our results indicate a limited percentage of large LACC variations in the S and O scenarios (about 10% and 20%, respectively). However, one can observe that in the W scenario over 40% of faults do severely impact the maximum LACC magnitude, hence representing the main source of overall system failures. In addition, as observed for the SA, the LACC's CDF for each **night** scenario behaves differently, according to the navigation features. More in detail, the considered faults in the S and W scenarios mostly produce minimal effects on the LACC (from about 75% to 84%, respectively). In contrast, in the O scenario, around 60% of the considered faults significantly corrupt the LACC – an effect that appears to be caused by the driving features of such a scenario. In fact, both SA and LACC show similar distributions of the maximum corruption impact in this scenario.

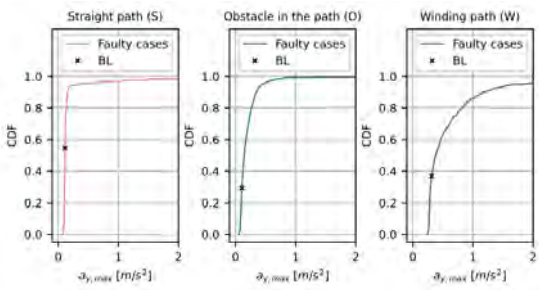
The results confirm that the probability that a fault in the perception system produces a major effect on the vehicle behavior depends on a combination of driving scenario and weather conditions. Specifically, the vehicle dynamics are strongly affected by faults in the S and O scenarios under **night** and **rainy** conditions, indicating that fault-vulnerable structures in perception may substantially impact the system performance. A broad overview of all results indeed shows that perception in the O scenario can be up to 5× more susceptible to faults than others, leading to much larger deviations in LACC and SA than the S and W scenarios.

For illustration purposes, Figure 11 depicts the temporal behavior of the fault-free baseline system (BL), and that of the system affected by a high-impact fault (FI) in terms of SA, LACC, and lane gap offset. As observed, at time 6.75 s, the



(a)

(b)



(c)

Fig. 10. CDF of Maximum lateral acceleration ($a_{y,max}$) for the daylight (a), night (b), and rainy (c) weather conditions on each evaluated dynamic scenario (S, O, W). Fault-free baseline (BL) values for daylight condition in S, O, and W scenarios are 0.09, 0.06, and 0.27 m/s^2, respectively. Baseline values for night conditions in S, O, and W scenarios are 0.1, 0.09, and 0.32 m/s^2, respectively. Baseline values for rainy conditions in S, O, and W scenarios are 0.11, 0.11, and 0.31 m/s^2, respectively.

.

fault affecting the perception causes a high lane gap offset error (right), which also causes effects on the SA (left) and LACC (center). At this point, the controller attempts to compensate for the error, resulting in a sudden high SA steering wheel angle (45 times higher w.r.t. the baseline value) and LACC (114 times higher w.r.t. the baseline value). The most critical impact in road lane gap offset (about 15 times) is reached at time 9.15 s. Such a deviation from the reference road lane demonstrates the physical susceptibility of a propagated fault in the vehicle behavior. It is also interesting to note that in this case, the controller's continuous operation effectively mitigates the majority of the effects on the vehicle's trajectory.

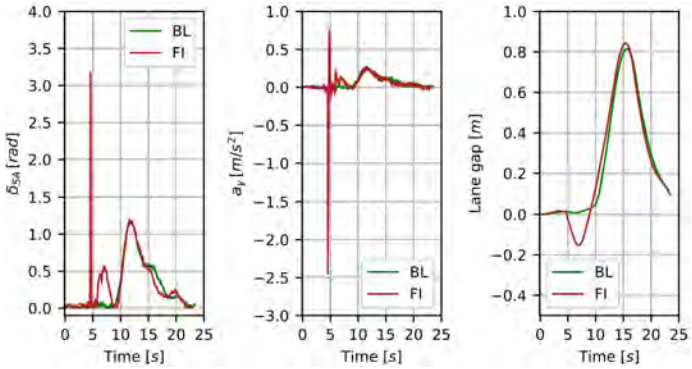


Fig. 11. Comparison of the temporal behavior between the fault-free baseline and a system affected by a high-impact fault in terms of steering wheel angle δ_{SW} (Left), lateral acceleration a_y (Center), lane gap offset (Right). **BL**: fault-free baseline. **FI**: faulty system.

Regarding the error sources, a deep analysis of the largest SA and LACC variations shows that the structures where faults may produce the most critical effects at the system level are part of the *Head* layers (from 1% to 62%) independently from the driving condition (Fig. 1), making them the primary candidates for hardening. On the other hand, faults in the *Backbone* and *Neck* were mainly masked by the interaction of the controlling agent and the implicit resilience of the closed-loop system.

6.2.2 Emergency Lane Keeping Assist . This section describes the experimental results of the dynamic evaluation of the ELKA application when the high-impact faults (identified through the static analysis) arise in the perception of the system. In the experiments, the vehicle navigates at 60 km/h with an unintentional lateral drift of 0.3 m/s, which leads it to come close to the right lane boundary and activate the ELKA operation [13]. Indeed, the ELKA's goal is to provide driver safety by preventing unintentional lane departure by means of a corrective steering action. The ELKA controller operates on the control principle mentioned in Section 5.1. Unless the driver disables it, the ELKA controller is always on and is activated when the vehicle comes within the lateral offset of 0.5 m from the lane boundaries. The position of each front corner with respect to the lane borders can be calculated by using the lane center offset, vehicle track width, and lane width. The perception model computes this lateral offset, which then triggers the ELKA controller to produce a corrective steering angle.

We leverage the **high-impact** and **low-impact SDC faults** identified by the static analysis to conduct a series of dynamic experiments. Since ELKA addresses strict lane keeping during the vehicle's navigation, we computed the critical impact effects as large deviations from the road. As highlited in Table 1, among all experiments, a total of two **high-impact** SDC faults produced a lane change, while no lane changes took place due to any of the **low-impact** SDC faults.

Figure 12 shows the impact of two representative faults (a **high-impact** SDC, in red, and a **low-impact** SDC, in green) on the vehicle trajectory in daylight conditions on a straight, dry road (impacts for the same and other scenarios exhibit similar behavior). At time 6 s, the ELKA controller gets triggered as the lateral distance between the lane line and the front right corner of the vehicle drops below 0.5 m. The high-impact SDC experiment demonstrates that, due to an inaccurate lane center offset estimation by the perception model, the ELKA controller is unable to maneuver the vehicle back into the lane. Failure to correct the vehicle trajectory causes the vehicle to exit the lane, which results in an unsafe situation. On the other hand, the **low-impact** SDCs do not produce any significant inaccuracy in computing the lane center offset. Due to the timely action of the ELKA controller acting on the steering angle, the vehicle is able to remain within its current lane.

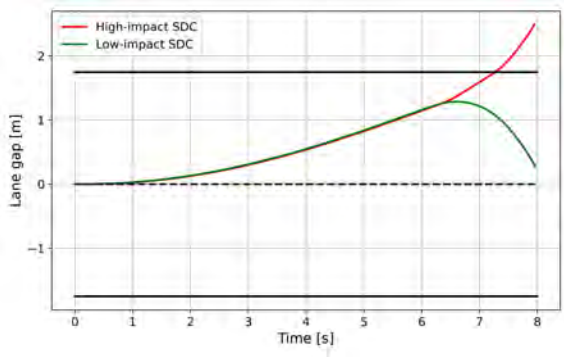


Fig. 12. Comparison of the impacts in road gap deviation for one of the high- and low-impact SDC faults for the Emergency Lane Keeping Assist (ELKA) application in daylight conditions. Red line: vehicle trajectory when a high-impact SDC fault arises. Green line: vehicle trajectory when a low-impact SDC fault arises. Black solid lines: lane boundaries. Black dashed line: lane center line.

Table 1. Comparison of KPI (lane changes) for low- and high-impact SDCs in daylight conditions

Fault set	Lane changes
Low-impact SDCs	0
High-impact SDCs	2

By confirming the effects on the system behavior of high- and low-impact SDC faults, these results demonstrate the generality of our proposed methodology for early reliability evaluation of automotive systems.

6.3 Overall Impact of Faults on the Automotive System

To evaluate the overall impact of faults affecting perception on automotive driving behavior, we classify and rank their effects on each driving scenario according to the navigation performance indicators. Since the impact of faults in some cases is null or negligible (as shown in the previous sections), we define a criticality threshold for each indicator.

Table 2 summarizes the quantitative performance results in each scenario and weather condition, considering the most critical fault propagation effects on the system for the LCA application. It is worth noting that equivalent trends were observed for the lateral deviation in ELKA application . Indeed, we identified the most critical cases that compelled the vehicle to move out of the current (desired) lane. First, we identified the fault propagation effects in the system causing significant impacts, including failures in a_{eq} greater than $0.315\,m/s^2$ (above this threshold, the driver senses discomfort as per the standards). Similarly, variations in LACC and SA greater than 1.5 times the fault-free baseline are considered significantly impactful, and their effect can thus be classified as a critical failure. In particular, our analyses indicate that for SA, LACC, and the equivalent acceleration, the **night** scenario with an obstacle (0) yielded a large number of high-impacting corruptions, as reported in Table 2. In fact, the most critical effects are observed in the 0 scenario. This is because the obstacles caused the lane markers to be inadequate, which resulted in an inaccurate lane center estimation. In addition, we computed the total number of critical system failures aiming to cause lane changes, as reported in the last column in Table 2. Interestingly, our results indicate that the weather condition has a stronger correlation with system failures than the complexity of a driving scenario. Considering **daylight** and **night** scenarios, we identified two and three collapsing lane change events independently of the driving scenario, respectively. Furthermore, all **rainy** scenarios were associated with up to nine collapsing lane change events.

Table 2. Comparison and rank of driving scenarios and weather conditions according to critical faults in the perception of the LCA application. Percentage distribution of high-impacting faults affecting the lateral acceleration (a_y), steering wheel angle (δ_{SA}), and equivalent acceleration (a_{eq})

Weather condition	Scenario	a_y (%)	δ_{SA} (%)	a_{eq} (%)	Lane changes (-)
Daylight	S	1.2	0.5	0.2	2
	W	6.5	0.5	0.4	
	O	44.4	12.6	1.8	
Night	S	2	0.9	0.3	3
	W	3.1	0.8	0.7	
	O	66	70.2	4.0	
Rainy	S	7.6	3.8	0.8	9
	W	39.7	2.3	1.2	
	O	44.8	35.1	0.6	

Table 3. Comparison and rank of driving scenarios and weather conditions according to results for the LCA application. Maximum observed corruptions on lateral acceleration (a_y), steering wheel angle (δ_{SA}), and equivalent acceleration (a_{eq})

Weather condition	Scenario	$a_{y,min}$ (m/s^2)	$a_{y,max}$ (m/s^2)	$\delta_{SA,min}$ (rad)	$\delta_{SA,max}$ (rad)	$a_{eq,min}$ (m/s^2)	$a_{eq,max}$ (m/s^2)
Daylight	S	0.07	5.72	0.25	8.02	0.01	0.96
	W	0.22	6.36	0.96	**9.42**	0.01	1.28
	O	0.02	5.67	0.05	**9.42**	0.006	0.96
Night	S	0.01	5.56	0.08	**9.42**	0.018	1.22
	W	0.15	**13.94**	0.88	**9.42**	0.015	1.47
	O	0.02	7.23	0.05	**9.42**	0.008	**1.68**
Rainy	S	0.07	5.82	0.26	**9.42**	0.04	1.28
	W	0.23	8.02	0.93	**9.42**	0.02	1.51
	O	0.05	6.85	0.11	**9.42**	0.02	1.03

For the first two cases (**daylight** and **night** scenarios), the corrupted perception could not accurately determine the lane center and propagated its effects across the system. Thus, the vehicle had to drive out of the reference lane due to this significant lane offset error, and the SA controller could not keep the vehicle in the lane. In contrast, the system failures in the **rainy** scenarios are primarily caused by the perception inability to precisely determine the lane center due to the combination of two factors: *i)* fault corrupting its expected functionality and *ii)* water droplets in the scenario that added disturbances to the system sensor (camera). These results indicate that 99.1% to 99.8% of the evaluated faults during the dynamic evaluation with TIARA were masked by the effect of the controller agent in the automotive system. However, the existence of about 0,2% to 0.9% of faults that caused critical system failures indicates the need to adopt complementary mechanisms to address and mitigate them.

In addition, we rank the overall quantitative impact of fault propagation effects on the LCA application and vehicle system for the nine evaluated scenarios in terms of maximum and minimum values of LACC, SA, and equivalent acceleration from the dynamic experiments, as reported in Table 3.

In general, the results show that a corrupted system causes similar deviations according to the driving scenario and performance indicators, with some exceptions associated with the combination of a driving scenario and an environmental condition, including the large deviations in LACC in **night** `W` scenario (around twice) and the slight increment in maximum equivalent acceleration in **night** `O` scenario. Interestingly, the maximum practical corruption impact in SA reached around 9.42 rad for almost all evaluated scenarios, which indicates that such a driving indicator is highly sensitive to corruption. Nevertheless, significant SA variations are insufficient to cause frequent collapse due to system failures, as previously discussed and reported in Table 2.

Overall, our experimental results indicate that from the system point of view, **daylight** `S` driving scenarios are remarkably resilient regarding critical impact effects and frequency of fault propagation at the system level. In fact, straight driving represents ideal vehicle navigation conditions, allowing the control agent to mitigate most effects.

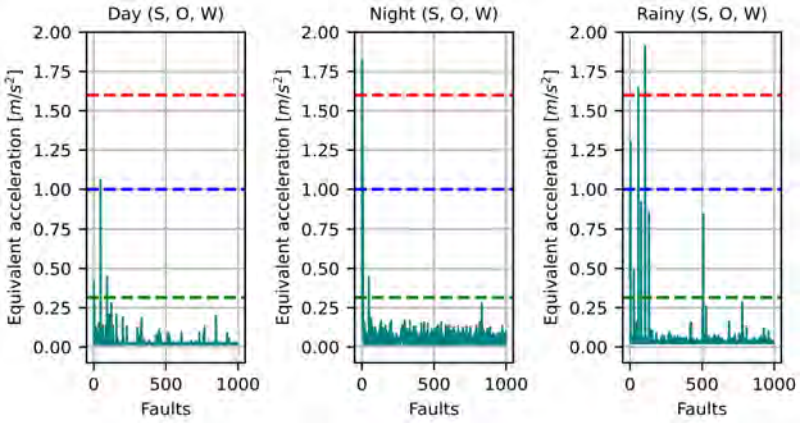


Fig. 13. Equivalent acceleration (Comfort metric) for the **daylight** (Left), **night** (Center), and **rainy** (Right) weather conditions on all scenarios (S, O, W) for the LCA application. Dashed green, dashed blue, and dashed red lines indicate comfortable, fairly uncomfortable, and uncomfortable, respectively.

Despite the high rates of significative fault propagation effects on **night** scenarios, as reported in Tables 2 and 3, suggest that such significant deviation magnitudes and frequencies of occurrence are not enough to treat the system's navigation. In these cases, another sub-system in the vehicle (controller) contributed to mitigate and remove their propagation effects. In contrast, **rainy** scenarios, especially `W`, are the most susceptible to producing critical impacts on the vehicle as a collapsing system failure through a lane change event.

Furthermore, we analyzed the equivalent acceleration as a comfort metric to determine the impact on the vehicle's passengers when the perception is faulty and propagates effects on the system, considering the top 1,000 high-impacting faults (in descending order) discovered in the static analysis. Figure 13 illustrates the maximum variations in equivalent acceleration for the evaluated faults on all driving scenarios in the three weather conditions for LCA. The three acceleration thresholds (*Comfortable*, *Fairly uncomfortable*, and *Uncomfortable*) indicate the critical operational limits a passenger might experience from a mild discomfort above 0.31 m/s^2, moderate discomfort above 1 m/s^2, and extreme discomfort above 1.6 m/s^2, as indicated by industry standards, such as ISO 2631-1 [31]. According to the results, most of the propagated fault effects on the vehicle were mainly masked by the action of the controller agent in the system, and the equivalent acceleration remained at acceptable comfort levels. However, due to faults in the perception model, many discomforts were perceived mainly in **rainy** conditions. In detail, the **night** and **rainy** cases show several corruption effects (from about 0.01% to 1.5% of evaluated faults) reaching (*Uncomfortable*) accelerations. In contrast, minimal corruption (about 0.002%) in the **daylight** case caused such critical accelerations, indicating that only in particular instances a corrupted perception might cause significative impacts on the passenger's comfort.

7 Verifying the Effectiveness of TIARA

To validate the effectiveness and main benefits of TIARA, we analyze two factors: *i)* the computational time required to perform the reliability evaluation, and *ii)* TIARA's ability to identify critical faults.

7.1 Computing Cost

We derive the computational cost of each step in TIARA for the two targeted applications, namely, LCA and ELKA.

Table 4. Average computational costs of each step of the TIARA approach for the nine evaluated scenarios

Step	Total frames per scenario	Total faults evaluated per scenario	Performance cost per evaluated fault (s)	Performance costs per scenario (h)	Overall performance costs of all experiments (h)
Static	3	7,940,846	1.0 (0.1*)	735.3 (73.4*)	6,617.4 (661.3*)
Dynamic	460	1,000	13.3	3.7	33.3

(*) Computational effort through distributed computing schemes.

The overall computational time of TIARA ($TIARA_{\text{Total}}$) is given by the computational cost of the static ($TIARA_S$) and dynamic ($TIARA_D$) steps, as follows.

$$TIARA_{\text{Total}} = TIARA_S + TIARA_D \tag{2}$$

$$TIARA_S = S \cdot WC \cdot AI_W \cdot A_{COST} \tag{3}$$

$$TIARA_D = S \cdot WC \cdot \sum_{i=1}^{k} SE_{COST} \tag{4}$$

where S is the total number of evaluated driving scenarios (e.g., road/traffic conditions), WC is the total number of evaluated weather conditions, AI_W is the number of weights in the AI-based perception application, A_{COST} is the average time cost to evaluate the effect of one injected fault in the AI-based application. Also, k is the total number of highly fault-vulnerable weights from the TIARA's static analysis, and SE_{COST} is the computational cost to inject a fault and execute the closed-loop system.

Table 4 presents the average execution costs for the LCA application during the static and dynamic evaluations by considering the number of analyzed video frames, the total amount of faults evaluated per driving scenario, and the overall costs for the nine considered scenarios ($S = 3$, $WC = 3$, $AI_W = 7.94 \cdot 10^6$, $k = 1,000$).

Initial performance tests allowed us to determine the characterization time, from 33.4 ms to around 1.0 s, on the static step (A_{COST}), which requires a moderate percentage (around 49%) of the time for model inference and about 42% of the time for the fitting and transformation processing steps. Indeed, for the characterization of the perception task, we used the worst-case complexity by adopting an exhaustive approach and evaluating all weights and layers in the application with a limited number of frames per scenario (3), which drastically increased the overall computational costs of the static step. Then, the dynamic evaluation experiments comprised virtual driving scenarios with a time length of 23.0 s, and 20 frames/s, which required the evaluation of 460 dynamic frames for each scenario. In detail, our experiments indicate that the TIARA framework required an average of 13.3 s ± 3.32 s time to compute and evaluate the closed-loop dynamic operation of the automotive system per scenario (SE_{COST}), resulting in about 3.7 h for the most critical 1,000 faults identified through the static analysis.

Then, the computational complexity of the static and dynamic analysis increases linearly with the number of evaluated faults per stage as:

$$TIARA_{S(AI_W)} \approx O(S \cdot WC \cdot AI_W \cdot A_{COST}) \approx S \cdot WC \cdot A_{COST} \cdot O(AI_W) \approx O(AI_W) \tag{5}$$

$$TIARA_{D(k)} \approx O(S \cdot WC \cdot k \cdot SE_{COST}) \approx S \cdot WC \cdot SE_{COST} \cdot O(k) \approx O(k) \tag{6}$$

Table 5. A comparison of KPIs after executing the dynamic evaluation step with TIARA for fault sets classified as SDCs and masked, during the static analysis step. All experiments targeted the S, W, O scenarios in daylight

Fault set	Scenario	$a_{y,rms}$ (m/s^2)	$a_{y,max}$ (m/s^2)	$\delta_{SA,rms}$ (rad)	$\delta_{SA,max}$ (rad)	$a_{eq,rms}$ (m/s^2)	$a_{eq,max}$ (m/s^2)
High-impact SDCs	S	1.14	5.72	1.54	8.03	0.08	0.96
	W	1.38	6.37	2.77	9.42	0.1	1.28
	O	1.61	5.67	1.63	9.42	0.14	0.96
Low-impact SDCs	S	0.04	0.13	0.17	0.41	0.03	0.05
	W	0.15	0.44	0.65	1.26	0.02	0.03
	O	0.07	0.43	0.05	0.37	0.02	0.04
Masked	S	0.04	0.14	0.18	0.4	0.03	0.06
	W	0.15	0.46	0.67	1.36	0.02	0.03
	O	0.08	0.37	0.05	0.24	0.02	0.04

Considering that the static step involves considering a number of faults much higher than the dynamic step, TIARA's computational complexity is dominated by the number of faults evaluated during the static step. We can therefore write:

$$TIARA_{\text{Total}} \approx O(AI_W) + O(k) \approx O(AI_W) \tag{7}$$

It is worth noting that TIARA can be used to target the complete universe of faults (e.g., exhaustive evaluation), or a subset of them (e.g., through custom or statistical fault injection strategies) to decrease the total number of faults to characterize [46, 50].

In addition, we estimate the reduction factor in time complexity allowed by TIARA (Static+Dynamic steps) against an exhaustive strategy for the closed-loop system evaluation. We compute the evaluation time for the exhaustive fault injection strategy (EX_{EVAL}) as,

$$EX_{EVAL} = S \cdot WC \cdot \sum_{i=1}^{n} SE_{COST} \tag{8}$$

where $S = 3$ is the number of scenarios, $WC = 3$ is the number of considered weather conditions, $n = 7.94 \cdot 10^6$ is the total number of weights in AI-based perception, and SE_{COST} is the execution cost to inject one fault and operate the vehicle. Assuming time costs per injected fault in the system ranging from ($SE_{COST} = [13.32s - 14.5s]$), we obtained execution times ranging from 264,430.17 h to 287,855.66 h for the overall characterization of the nine scenarios. Then a comparison between the overall exhaustive cost and the TIARA cost, from Table 4, indicates that TIARA can reduce the computational cost to exhaustively evaluate the closed-loop automotive system by 39.8 up to 42.3 times.

Next, we evaluate TIARA's effectiveness, i.e., how well it can identify the most critical faults and their effects. The results can then be used to identify the most suitable metrics to assess the system's vulnerability with acceptable confidence levels.

To evaluate TIARA's effectiveness, we assess the ability of the static step to identify the most critical faults, which are then considered for the dynamic analysis. To this end, we examine three different sets of faults affecting the layers of YoloP, each including 200 faults that were initially classified as SDCs (corruption effects) and masked (no corruption effects) during the static step. These fault sub-sets are as follows:

- **High-impact SDCs**: it includes faults classified as SDCs and selected for the dynamic evaluation as potentially critical causes of system failure.

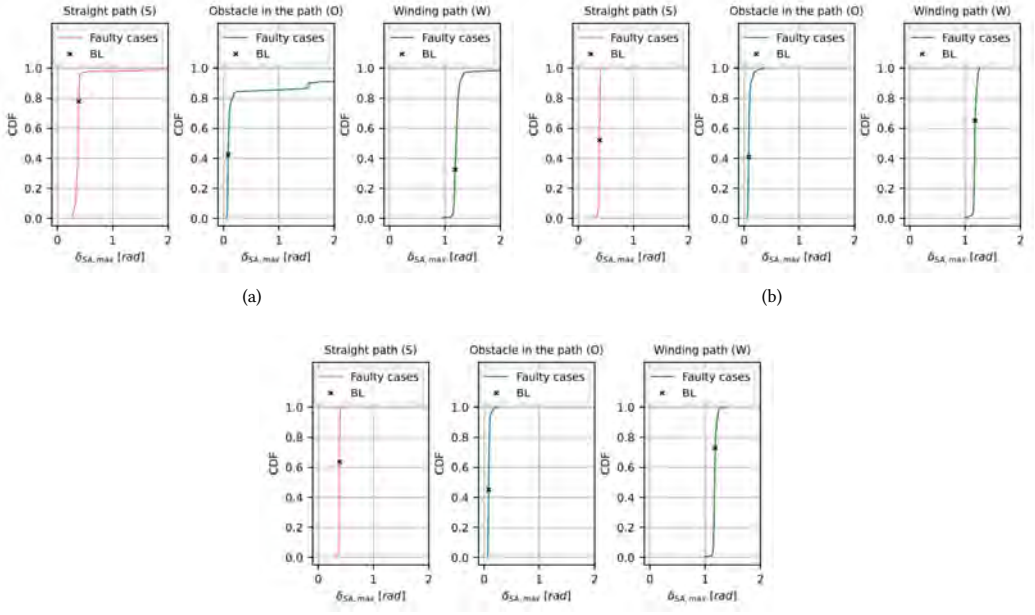


Fig. 14. Cumulative distribution function of maximum steering wheel angle ($\delta_{SA,max}$) due to **High-impact SDCs** (a), **Low-impact SDCs** (b), and **Masked** (c) per scenario (S, O, W) in **daylight**. Fault-free baseline (BL) values for daylight condition in S, O, and W scenarios are 0.39, 0.08, and 1.18 rad, respectively.

- **Low-impact SDCs**: it consists of faults, also classified as SDCs, which produced a low impact and were discarded for the dynamic evaluation.
- **Masked**: it includes faults affecting YoloP's layers that were identified as intrinsically resilient (namely, layers 17 to 33 in Figure 8).

Table 5 presents the *Root mean square* (*rms*) of LACC ($a_{y,rms}$), maximum LACC ($a_{y,max}$), *rms* of SA ($\delta_{SA,rms}$), maximum SA ($\delta_{SA,max}$), the rms of equivalent acceleration $a_{eq,rms}$, and maximum values $a_{eq,max}$ for the three fault sets after performing the dynamic evaluation through the TIARA strategy. We compute *rms* values to analyze the overall average behavior of the performance indicators during the experiments. Similarly, the maximum values are intended to underline the critical impacts caused by the propagation of a fault in the system. The results show that the *rms* and maximum values of magnitude for the evaluated KPIs are much higher for the faults that the static step labeled as high-impact SDCs than for those labeled as low-impact SDCs and masked. For all scenarios, we observed that low-impact SDCs show features similar to the masked faults, indicating that both sets of faults are highly unlikely to cause critical failures during the system's closed-loop operation. More in detail, Figure 14 illustrates the CDF of the maximum SA for the three fault sets (**High-impact SDCs**, **Low-impact SDCs**, and **Masked**) for all driving scenarios in daylight. The high-impact SDCs results show moderate deviations of SA magnitude for some scenarios. Moreover, critical variations in SA magnitude are observed for the O scenario. In contrast, the SA's CDF for the low-impact SDCs and masked faults produced a similar distribution of effects with minimal critical variations regarding its magnitude, with most of the variations near the fault-free baseline, confirming that the masked and low-impact faults barely propagate and cause significant errors in the automotive application.

Table 6. A comparison of the effectiveness and computational cost between TIARA and a statistical sampling strategy on the S, W, O scenarios under daylight conditions

Evaluation Strategy	Scenario	Targeted layers for fault injection	Overall Execution time of system evaluation (h)	Observed critical faults effects (%)	
				SA	LACC
Statistical	S	All	56.4	0.21	0.17
	W			55.23	0.18
	O			3.29	1.53
TIARA	S	Only the most critical (0 - 14 and 34 - 44)	33.2	0.24	0.18
	W			58.92	0.18
	O			3.37	1.71

These results suggest that both fault sets (low-impact SDCs and masked) can be neglected from further experiments involving the early reliability assessment of the system due to their minimal contribution to determining critical failures or identifying sensitive structures in perception. Furthermore, experimental results support TIARA's key idea that the static analysis effectively identifies those faults that are more prone to propagating their effects on the system; hence, their impact on the system's execution deserves further examination.

7.2 TIARA's Ability to Identify Most Critical Faults

We now focus on the closed-loop evaluation of the LCA application and compare the results of TIARA against a purely statistical-based reliability assessment strategy [50]. To this end, we consider a purely statistical strategy using a PytorchFI framework with the same fault model as TIARA (e.g., bit-flips in the weights). We inject a fault in a randomly selected YoloP weight, and observe its impact on the system behavior. Specifically, we conducted two fault injection campaigns: one based on statistical sampling using PyTorchFI and one with the TIARA framework, targeting all layers in YoloP. In both cases, we injected 1,692 faults with a 95% of confidence interval and 1.1% of error margin [37]. Each fault injection experiment required around 14.5 s with both strategies.

Table 6 presents, for both strategies, the impacts on two performance indicators (SA and LACC) for the LCA application, for the three considered scenarios (S, W, and O) in daylight. Moreover, we indicate the targeted layers per strategy, and the overall execution time per fault injection campaign on the system. It is worth noting that TIARA takes advantage of the static analysis to identify the most vulnerable layers in YoloP, so that the framework only injects faults into the most vulnerable ones (from 0 to 14 and 34 to 44), while others (around 41% of the evaluated faults) are directly classified as masked. This significantly reduces the overall computational time.

For the SA, the difference in percentage for the observed large magnitude (critical) faults between the two strategies is up to 14%. Similarly, a higher percentage of offset deviations can be observed in LACC, with a difference up to 11.7%. The SA and LACC results highlight that TIARA is able to effectively identify the critical faults, i.e., those that produce higher effects on the lateral vehicle behavior.

8 Validation of TIARA through Hardware-in-the-Loop Experiments

To validate the versatility of the TIARA strategy, we have implemented a HIL framework that allows the interaction of the controllers with the driving system and the evaluation of the effects of possible faults. The platform thus provides insights into how such faults would affect a real-world system. As depicted in Figure 15, our HIL setup is composed of a lane-centering controller, the perception model, and SCANeR studio that simulates the driving scenario, the vehicle, and the camera sensor.

The perception model is deployed on a Jetson AGX Xavier board. It is implemented using the AI Vehicle Computer RSL A3 embedded system from Syslogic AG, which combines an NVIDIA Jetson AGX Xavier OEM module with an integrated aluminum housing. Designed for high-performance edge AI processing, it is a 512-core Volta GPU with 64 Tensor Cores, an 8-core NVIDIA Carmel ARMv8.2 64-bit CPU, and 32GB 256-Bit LPDDR4x RAM [21].

To implement the lane centering controller, we use a Speedgoat's real-time machine. Speedgoat for Hardware-in-the-Loop (HIL) testing provides a real-time simulation environment that works seamlessly with Simulink for the development and validation of embedded control systems [54]. The Speedgoat's Baseline real-time target machine is a robust real-time simulation platform powered by an Intel Celeron 2.0 GHz quad-core processor, 4 GB RAM, and 256 GB SSD. It features an IO691 dual-channel CAN module (M12 connection) that supports standard CAN protocols using Simulink Real-Time. The standard interfaces include 4 Gbit Ethernet connections (three external RJ45), two RS232 interfaces, USB, and DisplayPort. The simulink model of the lane-centering controller is built and deployed into a real-time application. PyTorch Inference runs the YOLOP model directly on the Jetson AGX Xavier, using the PyTorch framework. SCANeR Studio, Jetson AGX Xavier, and the Speedgoat real-time system communicate over an Ethernet network, which uses the UDP protocol for fast, low-latency transfer of data. The SCANeR Studio simulator, installed in a workstation, produces a 1280×720p camera frame that is sent to the Jetson AGX Xavier. A perception model processes the image to determine the lane center offset error. The system can simulate a given scenario with or without faults (injected in the perception system). The effects of each injected fault are transmitted to the Speedgoat real-time target machine, which employs a control algorithm to determine the corresponding steering wheel angle required to maintain lane centering. Both the controller unit and the perception model run at a frequency of 10 Hz.

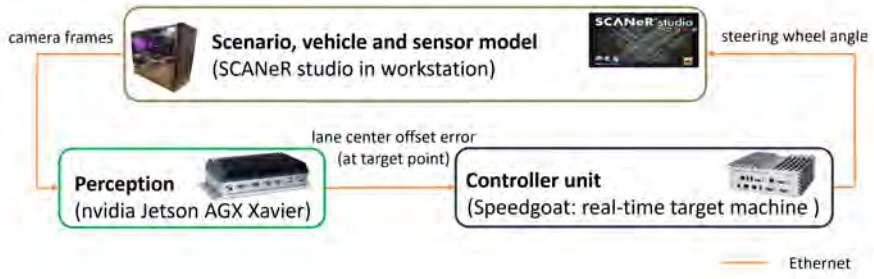


Fig. 15. Block diagram of the HIL platform.

Table 7. Comparison of KPIs (lateral acceleration, steering wheel angle, and lane changes) produced by TIARA and by the HIL setup for faults classified as low- and high-impact SDCs in the S, W, and O scenarios in daylight conditions

Fault set	$a_{y,rms}$ (m/s^2)		$\delta_{SA,rms}$ (rad)		No. of lane changes	
	Simulation	HIL	Simulation	HIL	Simulation	HIL
Low-impact SDCs	0.15	0.15	0.52	0.56	0	0
High-impact SDCs	0.18	0.19	0.53	0.63	2	2

The HIL tests of TIARA were set up for the S, W, and O scenarios in daylight, and the steering wheel angle and lateral acceleration were taken into account as KPIs. Ten experiments were carried out, considering the top 5 **low-impact** SDCs and 5 **high-impact** SDCs (as selected by the static phase), with the goal of comparing TIARA and HIL KPIs. An evaluation time of 14.5 s was adopted for each experiment, since the HIL environment used the real-time features in SCANeR studio.

Table 7 presents the average *rms* of LACC ($a_{y,rms}$) and the average *rms* of SA ($\delta_{SA,rms}$) for the low- and high-impact SDCs in simulation and HIL testing, as well as the number of occurring lane changes. Both low- and high-impact SDCs yield similar results in HIL compared to TIARA for all KPIs. Remarkably, we obtained exactly the same number of lane changes in simulation and in HIL testing. In summary, the results of the experimental testing demonstrate the effectiveness of TIARA's static analysis in identifying faults that show up in real-world scenarios, thus validating our approach.

9 Discussion

The findings of this work suggest that the TIARA methodology can effectively support the reliability analysis of AI-based automotive applications, such as perception, and contribute to the early reliability estimation of automotive systems at an affordable computational cost.

The results show that TIARA can effectively identify intrinsically resilient layers in perception for the two analyzed lane detection applications, i.e., Lane Centering Assistance and Emergency Lane Keeping Assist. Complementary evaluations of such structures through dynamic closed-loop assessment of the system further proved their minimal impact on the propagation of fault effects across the system. This highlights the importance of performing early reliability assessments on AI-based structures and understanding the impact of faults in closed-loop automotive systems, as they are pivotal to design improvements and to the development of fault countermeasure mechanisms.

In terms of performance, the TIARA strategy can reduce by up to 43.2 times the time required to perform an exhaustive evaluation of the automotive system. Clearly, our thorough fault analysis of the structures in the AI-based perception, through TIARA's static analysis, required considerable computational time (Table 4). However, we highlight that we used an exhaustive approach targeting all weights and layers in the application. In practice, TIARA's computing cost can be greatly decreased by about 10× by using distributed and multi-threading computing. Additional strategies to reduce computational costs, while preserving accuracy, include custom structural analyses or statistically-based fault injection. A complementary evaluation of TIARA highlighted its ability to ensure a very good effectiveness in identifying the most critical faults when compared with a standard statistically-based fault injection strategy for the closed-loop system evaluation. It is worth noting that TIARA's flexibility can be used to evaluate custom error models addressing features on the AI-based applications, such as errors arising on feature maps from hardware-aware faults.

In the dynamic evaluation of TIARA, we observed that selecting those faults that affected KPIs the most, in the static analysis, allows a swift identification and classification of failure events (e.g., lane change) for both the considered applications. Notably, our results show that around 0.2% to 0.9% of all evaluated cases can lead to critical system failures. Importantly, validation of TIARA through a Hardware-In-the-Loop platform indicated equivalent and consistent trends in the results.

Looking at the effect of corruption on relevant KPIs, the high rates of significative fault propagation on **night** scenarios (Tables 2 and 3) suggest that significant deviation in magnitudes and frequencies of occurrence are not directly associated with treats on the system's navigation. In fact, the proportional lateral controller agent can greatly mitigate the propagation effects in up to 99.8% of the evaluated cases.

According to our results, a combination of driving scenarios with obstacles `O` and winding curves `W`, during **night** and **rainy** conditions, are more prone to cause critical system failures than other road/traffic scenarios. Consequently, examining passenger discomfort through industry-level standard discomfort thresholds, as illustrated by Figure 13, provided an equivalent conclusion regarding the sensitivity of such scenarios to cause critical effects.

In the dynamic evaluation experiments with TIARA, we configured the vehicle with a low driving speed of 12 km/h and 60 km/h for LCA and ELKA, respectively that are preliminary yet representative velocities for autonomous shuttles commonly adopted on urban roads to estimate critical impact effects from a faulty AI-based application. Interestingly, such driving speeds allow us to analyze and identify the propagation effects of faults on the vehicle and identify collapsing system failures. According to our analyses, it is reasonable to expect that at higher vehicle speeds, the impact of a fault in AI applications might increase the occurrence of system failures.

Future research directions should consider evaluating more intricate controllers, sensors, and multi-model AI-based perception applications facing complementary challenges of real-time response operation and interacting with complementary driving applications.

10 Conclusions

This work introduced TIARA, a method to analyze the reliability characteristics of AI-based applications in automotive, studying the propagation effects of faults up to the system (i.e., vehicle) level. In order to tame the required computational effort, our method takes advantage of a two-step process. The first identifies the most vulnerable layers in the AI-based applications using exhaustive fault injection campaigns, while the second one performs fault injection campaigns focusing only on the faults affecting the most vulnerable layers (as identified by the first step), allowing the evaluation of their effects on the vehicle behavior in different scenarios.

The experimental results for the analysis of faults affecting the perception task in two case studies, corresponding to a Lane Centering Assistance and an Emergency Lane Keeping Assist application, indicate that TIARA can effectively identify the few faults producing significant impacts on the vehicle behavior, representing from around 0.2% to 0.9%, out of the vast universe of possible faults. Results produced by TIARA have also been validated through a hardware-in-the-loop implementation, showing TIARA's reliability in identifying the effects of faults on the vehicle's navigation.

A comparison of our evaluation strategy with a traditional exhaustive one demonstrated that TIARA can reduce by up to 43.2× the required computational time. In addition, a comparative evaluation between TIARA and a statistical-based fault sampling approach showed that, in spite of the dramatic reduction in computational cost, TIARA can preserve accuracy.

Results gathered using TIARA allow studying the link between the characteristics (e.g., in terms of road, traffic, and weather conditions) of a driving scenario and the impact of possible faults corrupting the perception task on the system behavior. The results in terms of three performance indicators, including equivalent acceleration associated with passenger comfort, indicate that rainy and night scenarios are more prone to generate collapsing system failures, such as lane changes, than daylight scenarios.

We emphasize that the proposed method allows for the evaluation of several traffic/road scenarios with minimal configuration changes, and can provide early-stage reliability estimation of the system components.

Future work can extend the analysis to other structures in automotive and IoT systems, including sensors and control agents, and explore the impact of other fault models on the system.

Acknowledgments

This study was supported by the EU Next-GenEU NRRP – M4C2, 1.4 – D.D. 1033 17/06/2022, CN00000023, within the MOST Project, by the National Resilience and Recovery Plan (PNRR) through the National Center for HPC, Big Data and Quantum Computing. Computational Resources were provided by HPC@PoliTo and by the project NODES, which has received funding from the MUR – M4C2 1.5 of PNRR with grant agreement no. ECS00000036.

This work reflects only the authors' views and opinions; neither the EU nor the EC is responsible for them.

References

[1] Mohammad Abboush, Daniel Bamal, Christoph Knieke, and Andreas Rausch. 2022. Hardware-in-the-Loop-Based Real-Time Fault Injection Framework for Dynamic Behavior Analysis of Automotive Software Systems. *Sensors* 22, 4 (2022). doi:10.3390/s22041360

[2] M. Abe and W. Manning. 2009. Chapter 5 - Steering System and Vehicle Dynamics. In *Vehicle Handling Dynamics*, M. Abe and W. Manning (Eds.). Butterworth-Heinemann, Oxford, 149–164. doi:10.1016/B978-1-85617-749-8.00005-2

[3] Jae-Gyung Ahn, Rhesa Nathanael, I-Ru Chen, Ping-Chin Yeh, and Jonathan Chang. 2021. Product Lifetime Estimation in 7nm with Large data of Failure Rate and Si-Based Thermal Coupling Model. In *2021 IEEE International Reliability Physics Symposium (IRPS)*. 1–6. doi:10.1109/IRPS46558.2021.9405193

[4] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. 2004. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing* 1, 1 (2004), 11–33. doi:10.1109/TDSC.2004.2

[5] AVSimulation. 2025. SCANeR - AVSimulation — avsimulation.com. https://www.avsimulation.com/en/scaner/. [Accessed 24-02-2025].

[6] Rahul Kumar Bhadani et al. 2018. The CAT Vehicle Testbed: A Simulator with Hardware in the Loop for Autonomous Vehicle Applications. *Electron. Proc. Theor. Comput. Sci.* 269 (April 2018), 32–47. doi:10.4204/eptcs.269.4

[7] Juan Borrego-Carazo, David Castells-Rufas, Ernesto Biempica, and Jordi Carrabina. 2020. Resource-Constrained Machine Learning for ADAS: A Systematic Review. *IEEE Access* 8 (2020), 40573–40598. doi:10.1109/ACCESS.2020.2976513

[8] Dennis Bruggner, Anoosh Hegde, Flavia Sofia Acerbo, Dhiraj Gulati, and Tong Duy Son. 2021. Model in the Loop Testing and Validation of Embedded Autonomous Driving Algorithms. In *2021 IEEE Intelligent Vehicles Symposium (IV)*. 136–141. doi:10.1109/IV48863.2021.9575530

[9] Lorenzo Brunelli, Alessandro Capancioni, Pierpaolo Gonnella, Rebecca Casadio, Alessandro Brusa, Nicolò Cavina, and Michele Caggiano. 2021. A Hybrid Vehicle Hardware-in-the-Loop System With Integrated Connectivity for Ehorizon Functions Validation. *IEEE Transactions on Vehicular Technology* 70, 5 (2021), 4340–4352. doi:10.1109/TVT.2021.3073807

[10] Long Chen, Yuchen Li, Chao Huang, Yang Xing, Daxin Tian, Li Li, Zhongxu Hu, Siyu Teng, Chen Lv, Jinjun Wang, Dongpu Cao, Nanning Zheng, and Fei-Yue Wang. 2023. Milestones in Autonomous Driving and Intelligent Vehicles—Part I: Control, Computing System Design, Communication, HD Map, Testing, and Human Behaviors. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 53, 9 (2023), 5831–5847. doi:10.1109/TSMC.2023.3276218

[11] Mohammad Cheshfar, Mohammad Hossein Maghami, Parviz Amiri, Hossein Gharaee Garakani, and Luciano Lavagno. 2024. Comparative Survey of Embedded System Implementations of Convolutional Neural Networks in Autonomous Cars Applications. *IEEE Access* 12 (2024), 182410–182437. doi:10.1109/ACCESS.2024.3510677

[12] Comma.ai. 2017. Openpilot. https://github.com/commaai/openpilot 24/10/2024.

[13] European Commission. 2021. Implementing regulation - 2021/646 - EN - EUR-Lex. https://eur-lex.europa.eu/eli/reg_impl/2021/646/oj/eng Doc ID: 32021R0646 Doc Sector: 3 Doc Title: Commission Implementing Regulation (EU) 2021/646 of 19 April 2021 laying down rules for the application of Regulation (EU) 2019/2144 of the European Parliament and of the Council as regards uniform procedures and technical specifications for the type-approval of motor vehicles with regard to their emergency lane-keeping systems (ELKS) (Text with EEA relevance) Doc Type: R Usr_lan: en.

[14] Josie E. Rodriguez Condia, Fernando Fernandes dos Santos, Matteo Sonza Reorda, and Paolo Rech. 2021. Combining Architectural Simulation and Software Fault Injection for a Fast and Accurate CNNs Reliability Evaluation on GPUs. In *2021 IEEE 39th VLSI Test Symposium (VTS)*. 1–7. doi:10.1109/VTS50974.2021.9441044

[15] Josie E. Rodriguez Condia, Juan-David Guerrero-Balaguera, Fernando F. Dos Santos, Matteo Sonza Reorda, and Paolo Rech. 2022. A Multi-level Approach to Evaluate the Impact of GPU Permanent Faults on CNN's Reliability. In *2022 IEEE International Test Conference (ITC)*. 278–287. doi:10.1109/ITC50671.2022.00036

[16] Bill Dally. 2023. Hardware for Deep Learning . In *IEEE Hot Chips 35 Symp. (HCS)*. 1–58. doi:10.1109/HCS59251.2023.10254716

[17] Pavan Kumar Datla Jagannadha, Mahmut Yilmaz, Milind Sonawane, Sailendra Chadalavada, Shantanu Sarangi, Bonita Bhaskaran, Shashank Bajpai, Venkat Abilash Reddy, Jayesh Pandey, and Sam Jiang. 2019. Special Session: In-System-Test (IST) Architecture for NVIDIA Drive-AGX Platforms. In *2019 IEEE 37th VLSI Test Symposium (VTS)*. 1–8. doi:10.1109/VTS.2019.8758636

[18] Adel Djoudi, Loic Coquelin, and Rémi Régnier. 2020. A simulation-based framework for functional testing of automated driving controllers. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. 1–6. doi:10.1109/ITSC45102.2020.9294454

[19] Alexey Dosovitskiy et al. 2017. CARLA: An Open Urban Driving Simulator. arXiv:1711.03938 [cs.LG] https://arxiv.org/abs/1711.03938

[20] Florian Faucher et al. 2022. SCANeR studio/SE-Workbench-RF physical radar sensor. *Proc. DSC Europe VR* (2022).

[21] Stefano Favelli, Meng Xie, and Andrea Tonoli. 2024. Sensor Fusion Method for Object Detection and Distance Estimation in Assisted Driving Applications. *Sensors* 24, 24 (Dec. 2024), 7895. doi:10.3390/s24247895 Publisher: MDPI AG.

[22] Mukul Anil Gosavi, Benjamin B. Rhoades, and James M. Conrad. 2018. Application of Functional Safety in Autonomous Vehicles Using ISO 26262 Standard: A Survey. In *SoutheastCon 2018*. 1–6. doi:10.1109/SECON.2018.8479057

[23] Juan-David Guerrero-Balaguera, Josie E. Rodriguez Condia, Fernando Fernandes Dos Santos, Matteo Sonza Reorda, and Paolo Rech. 2023. Understanding the Effects of Permanent Faults in GPU's Parallelism Management and Control Units. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '23)*. Article 46, 14 pages. doi:10.1145/3581784.3607086

[24] Said Hamdioui, Dimitris Gizopoulos, Groeseneken Guido, Michael Nicolaidis, Arnaud Grasset, and Philippe Bonnot. 2013. Reliability challenges of real-time systems in forthcoming technology nodes. In *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*. 129–134. doi:10.7873/DATE.2013.040

[25] Jin-Woo Han, M. Meyyappan, and Jungsik Kim. 2021. Single Event Hard Error due to Terrestrial Radiation. In *2021 IEEE International Reliability Physics Symposium (IRPS)*. 1–6. doi:10.1109/IRPS46558.2021.9405177

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37, 9 (2015), 1904–1916. doi:10.1109/TPAMI.2015.2389824

[27] Shailesh Sudhakara Hegde et al. 2025. Estimating the Impact of Soft Errors on AI-based Perception in Automotive. In *Latin american Test Symposium (LATS'25)*. 1–6, to appear.

[28] Yijie Hou, Chengshun Wang, Junhong Wang, Xiangyang Xue, Xiaolong Luke Zhang, Jun Zhu, Dongliang Wang, and Siming Chen. 2022. Visual Evaluation for Autonomous Driving. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (2022), 1030–1039. doi:10.1109/TVCG.2021.3114777

[29] Saurabh Hukerikar, Atieh Lotfi, Yanxiang Huang, Jason Campbell, and Nirmal Saxena. 2024. Optimizing Large-Scale Fault Injection Experiments through Martingale Hypothesis: A Systematic Approach for Reliability Assessment of Safety-Critical Systems. In *54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S)*. 111–117.

[30] IEEE. 2022. THE INTERNATIONAL ROADMAP FOR DEVICES AND SYSTEMS: 2022. In *Institute of Electrical and Electronics Engineers (IEEE)*.

[31] International Organization for Standardization. 1997. ISO 2631-1:1997 - Mechanical vibration and shock — Evaluation of human exposure to whole-body vibration.

[32] International Organization for Standardization. 2011. BS ISO 8855:2011 - Road vehicles — Vehicle dynamics and road-holding ability — Vocabulary.

[33] International Organization for Standardization. 2022. BS ISO 34502:2022 - Road vehicles — Test scenarios for automated driving systems — Scenario-based safety evaluation framework.

[34] Saurabh Jha et al. 2019. Kayotee: A Fault Injection-based System to Assess the Safety and Reliability of Autonomous Vehicles to Faults and Errors. arXiv:1907.01024 [cs.SE] https://arxiv.org/abs/1907.01024

[35] Saurabh Jha, Subho S. Banerjee, James Cyriac, Zbigniew T. Kalbarczyk, and Ravishankar K. Iyer. 2018. AVFI: Fault Injection for Autonomous Vehicles. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. 55–56. doi:10.1109/DSN-W.2018.00027

[36] Chen Jiacheng, Zhou Haibo, Zhang Ning, Yang Peng, Gui Lin, and Shen Xuemin Sherman. 2016. Software Defined Internet of Vehicles: Architecture, Challenges and Solutions. *Journal of Communications and Information Networks* 1, 1 (2016), 14–26. doi:10.11959/j.issn.2096-1081.2016.002

[37] R. Leveugle, A. Calvez, P. Maistri, and P. Vanhauwaert. 2009. Statistical fault injection: Quantified error and confidence. In *2009 Design, Automation Test in Europe Conference Exhibition*. 502–506. doi:10.1109/DATE.2009.5090716

[38] Chenxi Liu, Hao Yang, Meixin Zhu, Feilong Wang, Torgeir Vaa, and Yinhai Wang. 2024. Real-Time Multi-Task Environmental Perception System for Traffic Safety Empowered by Edge Artificial Intelligence. *IEEE Transactions on Intelligent Transportation Systems* 25, 1 (2024), 517–531. doi:10.1109/TITS.2023.3309100

[39] Zongwei Liu, Wang Zhang, and Fuquan Zhao. 2022. Impact, challenges and prospect of software-defined vehicles. *Automotive Innovation* 5, 2 (2022), 180–194. doi:10.1007/s42154-022-00179-z

[40] Sara Luciani, Angelo Bonfitto, Nicola Amati, and Andrea Tonoli. 2020. Model predictive control for comfort optimization in assisted and driverless vehicles. *Advances in Mechanical Engineering* 12, 11 (2020), 1687814020974532. doi:10.1177/1687814020974532

[41] Abdulrahman Mahmoud, Neeraj Aggarwal, Alex Nobbe, Jose Rodrigo Sanchez Vicarte, Sarita V. Adve, Christopher W. Fletcher, Iuri Frosio, and Siva Kumar Sastry Hari. 2020. PyTorchFI: A Runtime Perturbation Tool for DNNs. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. 25–31. doi:10.1109/DSN-W50199.2020.00014

[42] United Nations. 2014. Regulation No 130 of the Economic Commission for Europe of the United Nations (UN/ECE) — Uniform provisions concerning the approval of motor vehicles with regard to the Lane Departure Warning System (LDWS). 29-46 pages.

[43] Huy-Hung Nguyen, Duong Nguyen-Ngoc Tran, Long Hoang Pham, and Jae Wook Jeon. 2024. Optimizing Monocular Driving Assistance for Real-Time Processing on Jetson AGX Xavier. *IEEE Access* 12 (2024), 71853–71865. doi:10.1109/ACCESS.2024.3402239

[44] Thang Nguyen and Stuart Wooters. 2014. FPGA-based development for sophisticated automotive embedded safety critical system. *SAE Int. J. Passeng.* 7, 2014-01-0240 (2014), 125–132. doi:10.4271/2014-01-0240

[45] Nvidia Corporation. 2023. isaacsim. https://docs.omniverse.nvidia.com/isaacsim/latest/index.html 24/10/2024.

[46] Francesco Pessia, Juan-David Guerrero-Balaguera, Robert Limas Sierra, Josie E. Rodriguez Condia, Marco Levorato, and Matteo Sonza Reorda. 2024. Effective Application-level Error Modeling of Permanent Faults on AI Accelerators. In *2024 IEEE 30th International Symposium on On-Line Testing and Robust System Design (IOLTS)*. 1–7. doi:10.1109/IOLTS60994.2024.10616087

[47] Henrich C. Pöhls. 2023. Towards a Unified Abstract Architecture to Coherently and Generically Describe Security Goals and Risks of AI Systems. In *Security and Trust Management*, Ruben Rios and Joachim Posegga (Eds.). 85–94. doi:doi.org/10.1007/978-3-031-47198-8_5

[48] C. Prasad, L. Jiang, D. Singh, M. Agostinelli, C. Auth, P. Bai, T. Eiles, J. Hicks, C. H. Jan, K. Mistry, S. Natarajan, B. Niu, P. Packan, D. Pantuso, I. Post, S. Ramey, A. Schmitz, B. Sell, S. Suthram, J. Thomas, C. Tsai, and P. Vandervoorn. 2013. Self-heat reliability considerations on Intel's 22nm Tri-Gate technology. In *2013 IEEE International Reliability Physics Symposium (IRPS)*. 5D.1.1–5D.1.5. doi:10.1109/IRPS.2013.6532036

[49] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. arXiv:1506.02640 [cs.CV] https://arxiv.org/abs/1506.02640

[50] A. Ruospo, G. Gavarini, C. de Sio, J. Guerrero, L. Sterpone, M. Sonza Reorda, E. Sanchez, R. Mariani, J. Aribido, and J. Athavale. 2023. Assessing Convolutional Neural Networks Reliability through Statistical Fault Injections. In *2023 Design, Automation Test in Europe Conference Exhibition (DATE)*. 1–6. doi:10.23919/DATE56975.2023.10136998

[51] Sven Rzepka, Alexander Otto, Dietmar Vogel, and Rainer Dudek. 2018. Application-Driven Reliability Research of Next Generation for Automotive Electronics: Challenges and Approaches. *Journal of Electronic Packaging* 140, 1 (03 2018), 010903. doi:10.1115/1.4039333

[52] Fernando Fernandes dos Santos, Pedro Foletto Pimenta, Caio Lunardi, Lucas Draghetti, Luigi Carro, David Kaeli, and Paolo Rech. 2019. Analyzing and Increasing the Reliability of Convolutional Neural Networks on GPUs. *IEEE Transactions on Reliability* 68, 2 (2019), 663–677. doi:10.1109/TR.2018.2878387

[53] Dinesh Cyril Selvaraj, Shailesh Hegde, Carla Fabiana Chiasserini, Nicola Amati, Francesco Deflorio, and Giuliana Zennaro. 2021. A Full-fledge Simulation Framework for the Assessment of Connected Cars. *Transportation Research Procedia* 52 (2021), 315–322. doi:10.1016/j.trpro.2021.01.037 23rd EURO Working Group on Transportation Meeting, EWGT 2020, 16-18 September 2020, Paphos, Cyprus.

[54] Speedgoat. 2025. Baseline real-time target machine | Speedgoat. https://www.speedgoat.com/products-services/real-time-target-machines/baseline-real-time-target-machine [Accessed 07-07-2025].

[55] Daniele Sportillo, Alexis Paljic, and Luciano Ojeda. 2019. On-Road Evaluation of Autonomous Driving Training. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. 182–190. doi:10.1109/HRI.2019.8673277

[56] Andrzej J. Strojwas, Kelvin Doong, and Dennis Ciplickas. 2019. Yield and Reliability Challenges at 7nm and Below. In *2019 Electron Devices Technology and Manufacturing Conference (EDTM)*. 179–181. doi:10.1109/EDTM.2019.8731146

[57] Emil Talpes, Debjit Das Sarma, Ganesh Venkataramanan, Peter Bannon, Bill McGee, Benjamin Floering, Ankit Jalote, Christopher Hsiong, Sahil Arora, Atchyuth Gorti, and Gagandeep S. Sachdev. 2020. Compute Solution for Tesla's Full Self-Driving Computer. *IEEE Micro* 40, 2 (2020), 25–35. doi:10.1109/MM.2020.2975764

[58] João P. Trovao. 2019. Trends in Automotive Electronics [Automotive Electronics]. *IEEE Vehicular Technology Magazine* 14, 4 (2019), 100–109. doi:10.1109/MVT.2019.2939757

[59] Timothy Tsai, Siva Kumar Sastry Hari, Michael Sullivan, Oreste Villa, and Stephen W. Keckler. 2021. NVBitFI: Dynamic Fault Injection for GPUs. In *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 284–291. doi:10.1109/DSN48987.2021.00041

[60] Zejiang Wang, Jingqiang Zha, and Junmin Wang. 2021. Autonomous Vehicle Trajectory Following: A Flatness Model Predictive Control Approach With Hardware-in-the-Loop Verification. *IEEE Transactions on Intelligent Transportation Systems* 22, 9 (2021), 5613–5623. doi:10.1109/TITS.2020.2987987

[61] Sijie Wei, Peter E. Pfeffer, and Johannes Edelmann. 2023. State of the Art: Ongoing Research in Assessment Methods for Lane Keeping Assistance Systems. *IEEE Transactions on Intelligent Vehicles* (2023), 1–28. doi:10.1109/TIV.2023.3269156

[62] Dong Wu, Man-Wen Liao, Wei-Tian Zhang, Xing-Gang Wang, Xiang Bai, Wen-Qing Cheng, and Wen-Yu Liu. 2022. YOLOP: You Only Look Once for Panoptic Driving Perception. *Machine Intelligence Research* 19, 6 (2022), 550–562. doi:10.1007/s11633-022-1339-y

[63] Xizhe Zhang, Siddartha Khastgir, Justin-Kiyoshi Tiele, Kazuhito Takenaka, Tasuku Hayakawa, and Paul Jennings. 2024. ODD and Behavior Based Scenario Generation for Automated Driving Systems. *IEEE Access* 12 (2024), 10652–10663. doi:10.1109/ACCESS.2024.3350512

Received 31 March 2025; revised XX XXXX 2025; accepted XX XXXX 2025