Frontiers in Computer Science and Artificial Intelligence
ISSN: 2978-8048
DOI: 10.32996/fcsai
Journal Homepage: www.al-kindipublisher.com/index.php/fcsai

FCSAI
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

| RESEARCH ARTICLE

AI-Assisted Embedded System Verification Framework for Connected Vehicles

Leela Lakshmi Sirana Jayaram
Embedded systems Engineer, Oakland University, Michigan, United States
Corresponding Author: Leela Lakshmi Sirana Jayaram, **E-mail**: ssjayaram86@gmail.com

| ABSTRACT

The inclusion of electronic control units (ECUs), sensors, communication networks, and connectivity services has added more complexity to the embedded systems of connected vehicles. Classical verification techniques often rely on pre-defined test cases and manual verification. However, when working with a large number of test cases, dynamic operating conditions, and multiple interactions between different subsystems of the vehicle, such verification techniques might not be efficient enough. In this regard, this paper suggests a new AI-based verification framework for increasing the efficiency and reliability of verification in connected vehicles. The suggested framework includes an AI-based test generation and prioritization process, verification environment for connected vehicles, continuous monitoring of the system, anomaly detection, and intelligent verification analytics in a closed-loop architecture. AI will be used in identifying critical test cases, detecting anomalies in sensors, ECUs, timing, and communication, as well as adaptive test selection depending on the results of the verification process. To show the practicality of the suggested framework, a simulation test case of connected vehicles will be used with some representative sensor, ECU, communication, and connectivity faults. The performance of the framework will be assessed by test coverage, fault detection rate, verification time, and false positives.

| KEYWORDS

AI, ECU, connected vehicles, connectivity faults

| ARTICLE INFORMATION

ACCEPTED: 02 August 2026 **PUBLISHED:** 08 September 2026 **DOI:** 10.32996/fcsai.2026.5.9.21

1. Introduction

Change is happening in the automotive sector due to the rapid development of smart and connected cars. The current vehicles use embedded electronics to perform tasks related to engine control, safety, driver assistance, comfort, diagnostics, and communication. Embedded electronics comprise various Electronic Control Units (ECUs), sensors, actuators, communication network, and embedded software which exchange information constantly to achieve efficient operation of the car. With the integration of different communication technologies like V2V, V2I, V2C, and others, more interaction is now available between the in-vehicle embedded system and communication environment [1][2].

The more the number of embedded functions grows, the harder it is to verify them. Traditional approaches to verification involve test cases, requirements-driven testing, simulations, and evaluation of system reactions to them. Despite the fact that all these techniques are still relevant for the development of vehicles, they might consume much time and effort if used to analyze a great number of operational modes and interactions between various system components. System failures can be caused by faulty sensors, communication problems, ECU reaction issues, timing errors, packet loss, or even interactions between car subsystems. It is hard to perform an exhaustive test of all such combinations using traditional techniques [3].

The capability of AI technology would help to enhance the verification process for embedded systems by assisting engineers to find the most important test cases and analyze the behavior of complex systems. Machine learning techniques and intelligent optimization algorithms can be used for verification data, system specifications, sensor signals, network data, and ECU responses.

Such tools could provide the opportunity for test-case generation, risk-based test selection, anomalies identification, failure prediction, and verification results analysis. With the help of AI technology, all possible test cases do not have to be tested equally; only the most important ones are chosen for verification [4].

The study has noticed that several existing approaches focus on individual verification tasks such as generation of test cases, detection of anomalies, or monitoring of communications. On the other hand, an approach which integrates all these aspects into a closed-loop verification can create more value for connected vehicles' applications. Therefore, this paper aims at developing an AI-enabled verification of embedded systems approach, which integrates intelligent test case generation and prioritization, connected vehicle simulation, monitoring of the system, detection of anomalies, analysis of the results of verification, and test optimization based on feedback. The applicability of the proposed approach will be illustrated via a simulated connected-vehicle application scenario.

2. Literature Review

Fast adoption of connected cars that utilize software-based technology has dramatically changed the architectural considerations as well as the verification demands of automotive embedded systems. Today's cars include numerous electronic control units (ECUs), sensors, actuators, communication network, and connectivity modules to enable powertrain control, safety features, driver assistance, diagnostics, infotainment, and vehicle connectivity. These components communicate large amounts of information over communication networks within the car such as Controller Area Network (CAN), Local Interconnect Network (LIN), and automotive Ethernet. In addition to this, connected cars expand the scope of interaction by introducing vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), and vehicle-to-cloud interaction. Therefore, the verification process should encompass not only verification of ECU functionalities but also verification of interactions between different elements [5].

The traditional verification of automotive embedded systems usually uses stages of testing via Model-in-the-Loop (MIL), Software-in-the-Loop (SIL), and Hardware-in-the-Loop (HIL). The MIL approach allows evaluating control approaches at the early stage of development, while the SIL method allows validating software implementation using simulation. In the case of HIL testing, actual controllers are tested in a simulated environment. In automotive software development, the following testing types are applied: requirements-driven testing, unit testing, integration testing, and regression testing. Despite the fact that these techniques allow systematic verification, they may not work effectively due to the increasing number of aspects that need to be taken into account [6].

The use of artificial intelligence is gaining popularity as one of the technologies used in automotive verification and testing. The machine learning process uses previous test results as well as behaviors of the system in order to detect patterns which relate to the failure cases and select those scenarios with high verification importance. Test generation using AI processes can generate various combinations of operational parameters which are hard to identify manually. Also, intelligent optimization processes can eliminate duplicate test cases through the selection of those test cases that consider risks, critical requirements, failures, and information gain expectations [7][8].

Another important use of AI technology involves anomaly and fault detection. Vehicles have sensors that give various measurements, responses from ECUs, diagnostics, timings, and network communications. AI algorithms can learn the patterns which define normal system operation and any anomalies that could imply a fault in the system. Examples of such anomalies could be irregular sensor measurements, unusual responses from ECUs, communication delays, unusual frequencies of messages on CAN buses, lost packets, and improper state transitions. It becomes especially helpful when the system's behavior cannot be described with simple fixed rules or thresholds [9][10].

As embedded systems communicate more frequently with outside networks and cloud services, they need additional checks to ensure that they function properly. Issues like latency, packet loss, unexpected disconnections, inconsistent data, and unforeseen network conditions may affect the operation of a vehicle. Thus, any verification process that needs to be performed on the vehicle will need to consider its functionality and communication abilities [11].

Although much advancement has been made in AI-aided car testing, existing techniques tend to emphasize individual tasks like test case creation, error estimation, anomalies identification, or network monitoring. Lack of interconnection between all these features may limit the potential of verification systems in making use of the errors discovered to enhance further testing. This calls for an integrated framework in which AI-aided test case generation, connected cars simulation, anomalies identification, verification analytics, and test optimization can work harmoniously together. The proposed framework aims at bridging this gap [12][13].

3. Proposed AI-Assisted Embedded System Verification Framework

The intended AI-aided verification framework for an embedded system for connected vehicles is set out to improve the efficiency, effectiveness, and coverage of the verification process for connected vehicles. The framework incorporates requirements for the

system, AI-aided test generation, simulation of the connected vehicle, monitoring of the embedded system, anomaly detection, verification analysis, and optimization based on feedback within a closed loop design process. This approach does not intend to replace existing processes of verification but utilizes AI to support decision making for engineers.

3.1 Framework Architecture

These layers include six main functions: input and requirements definition, test generation and prioritization through artificial intelligence, verification environment of a connected vehicle, system monitoring and anomaly detection, verification analysis, and test optimization based on feedback. Input information about the expected system behavior is derived from requirements and vehicle model. Based on this input, the AI module is responsible for generating and prioritizing test cases. These test cases are executed in a simulated connected vehicle environment which involves ECUs, sensors, actuators, communication and external connectivity systems. System behavior is monitored for anomalies throughout the process. Verification results are analyzed and provided as feedback for future test cases.

The general data flow is described as follows:

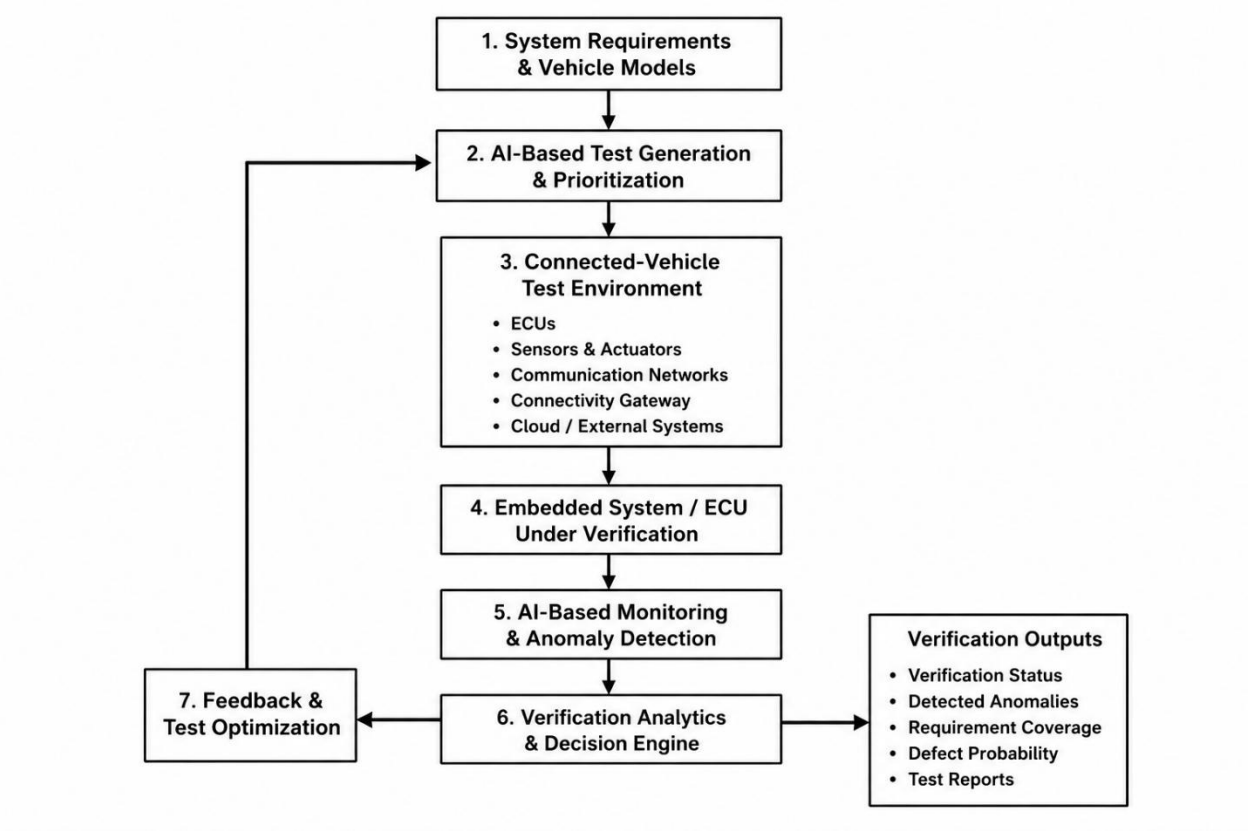

Fig 1. AI-assisted embedded system verification framework

The suggested verification framework for embedded systems assisted by AI works in a closed-loop fashion to improve verification for connected vehicles. To begin with, requirements of the system and the vehicle model are taken as an input for the AI-based test generation and prioritization module to detect relevant test cases which pose a high risk. Such test cases are performed in a connected vehicle setting which includes ECUs, sensors, actuators, communication network, connectivity gateway, and other external systems. The behavior of the embedded system or ECU under verification is constantly monitored using AI-based monitoring and anomaly detection which helps in detecting abnormal behavior, communication problems, or any faults in the system. The verification analytics and decision engine analyze the outcome and produce output that consists of verification status, anomalies detected, requirement coverage, probability of defects, and test reports. Finally, the outcomes are used to optimize the test scenarios using the test optimization module.

Such closed-loop design enables verification processes to adjust based on the past behavior of systems.

3.2 Input and Requirement Layer

This is where the data needed to define the expected behavior of the system and verification goals is provided. The inputs could include the functional requirements, specifications of the ECU, the operating range of the sensors, the communication needs, time constraints, diagnostic requirements, test results from previous tests, and known fault conditions. For example, one of the functional requirements would be the permissible response time of the ECU upon receiving a certain input from the sensor.

The information gathered is used to create verification criteria and constraints. The essential requirements receive more weight in order to make sure that the AI component can handle situations that may have a bigger effect on the performance of the system or the vehicle. Past instances of failure may be used for identifying requirements for extra verification.

3.3 AI-Based Test Generation and Prioritization

The AI-based test generation module generates or chooses scenarios for testing depending on the system requirements and characteristics. The test cases may involve various combinations of speed of the car, readings from sensors, operating modes of the ECU, delays in communications, packets losses, and the connectivity status. Instead of giving equal importance to all the test cases, a priority is set for each scenario.

The test priority score may be described conceptually as:

$$TP = f(R, FP, RC, HD) \quad \text{..... (1)}$$

In this way, R stands for the risk factor, FP stands for False positive (the expected likelihood of failure), RC stands for requirement criticality, and HD stands for the historical defects data. The test cases having high-priority score are conducted first or more frequently.

Also, the machine learning or optimization algorithm can detect those operating conditions that are not sufficiently tested. This method is aimed at reducing the redundant testing efforts and focusing on risky and non-tested conditions.

3.4 Connected-Vehicle Verification Environment

The chosen test scenarios are carried out in a connected vehicle simulation environment. In such an environment, certain components that include ECUs, sensors on board the vehicle, actuators, CAN communication protocol, gateway for communication and communication with the outside world or the cloud are modeled. Hence, different states and failure modes can be simulated without necessarily using a real car.

For example, different values for sensors, delayed CAN messages, missing packets in communication, incorrect response from the ECUs, or even temporary loss of cloud connectivity can be simulated. Such an architecture can then be used to create Software-in-the-Loop or Hardware-in-the-Loop verification environments.

3.5 AI-Based Monitoring and Anomaly Detection

The monitoring module continually collects data from sensors, ECUs, communications systems, and connectivity channels during the test run. These observations are analyzed in comparison with expected patterns of operation for anomalies.

An anomaly index, in theory, can be represented by:

$$AS = f(SD, TD, NB, ER) \quad \text{.....(2)}$$

Here, SD stands for sensor deviation, TD stands for timing deviation, NB stands for network behavior, and ER stands for ECU response. If the calculated anomaly score exceeds a certain threshold value, the corresponding test scenario is marked for further investigation.

The anomaly detection component can detect conditions such as sensor signal deviation, controller response time delay, unexpected CAN messages, disruption of communication, and system state transition errors. The use of artificial intelligence in monitoring the condition becomes especially effective in detecting intricate patterns that cannot be easily detected using threshold-based rules alone.

3.6 Verification Analytics and Decision Engine

This is accomplished by the verification analytics module that handles the outputs generated after the execution and anomaly identification. The scenario is classified in categories such as pass, caution, possible defect, and critical failure. The decision engine further gives details regarding the requirement that is being affected, the degree of anomaly, confidence, and verification action.

The key verification metrics include test coverage, fault detection ratio, time taken to verify, false positive rate, and test coverage of critical situations. Using these measures makes it possible for engineers to see whether there is any improvement brought by the suggested method compared to traditional verification techniques.

AI-based classification results are meant to help in making engineering decisions but not to automatically validate safety-critical functions of vehicles. Verification decisions are still dependent on the engineering review process.

3.7 Feedback-Based Test Optimization

An important component of the proposed model is the feedback loop between the result of verification and test generation. When a problem or anomaly is found, the details about the corresponding conditions are fed back to the AI program. The model then increases the priority of the similar cases, creates additional tests in the area of failure and reduces the unnecessary repetition of cases that work properly.

For example, where the communication delays result in an unusual response from the ECU at a particular operating condition, then other tests can be created by varying the delays and sensors used. This method allows the testing procedure to become gradually more concentrated in the weak parts of the operation.

3.8 Framework Output

The process comes to an end with the formation of an analytical verification report which includes the test status, anomalies found, requirements that are impacted, fault severity, test coverage, fault detection capability, and suggestions for testing. With the use of intelligent test scheduling, anomaly detection, verification analytics, and optimization through feedback, the process provides a method for the verification of embedded systems. Therefore, this framework has been designed in such a way that it would help with systematic verification of complex connected vehicle systems without performing unnecessary tests.

4. Simulated Validation Scenario

The use of a simulated connected-vehicle environment has been used to study the relevance and effectiveness of the proposed AI-enabled framework for verification of embedded systems. The purpose of such a simulation is to find out if using AI to prioritize and detect anomalies in testing would improve the verification process by improving fault detection as well as decreasing verification efforts in comparison with traditional testing techniques. The simulation includes typical interactions between vehicle ECUs, sensors, communications networks, and external connectivity without using a real vehicle or HIL setup.

4.1 Simulation Configuration

The simulated connected-vehicle system consists of five ECUs, twelve sensor inputs, Controller Area Network (CAN), connectivity gateway, and a vehicle-to-cloud communication interface. The different ECUs simulated various vehicle control functionalities and communicated through the CAN network using sensor and control data. The connectivity gateway provided an interface between the in-vehicle network and the external cloud system.

The number of verification cases that have been developed is 1,000, using a combination of varying inputs to the sensors, the state of operation of the ECU, conditions of communication, and connectivity parameters. A combination of normal and abnormal cases was used to test the effectiveness of the verification process in distinguishing between the two. In the AI-assisted method, the priority was given according to risk, requirement importance, and probability of abnormal behavior.

4.2 Fault Injection Scenarios

Selected faults were intentionally planted in the simulation in order to imitate possible problems that may occur in connected-vehicle embedded systems. Sensor faults were introduced via faulty, non-existent, or out-of-the-range sensor readings. Controller faults included wrong response of the controller or unexpected system states. Communication faults included delays in CAN messages, missed CAN messages, and abnormal CAN messages rates. Connectivity faults were introduced by dropping packets, higher communication delay, and momentary interruption of the cloud-vehicle communication.

Main fault categories used for validation are listed in Table 1.

Fault Category	Simulated Condition	Expected Verification Response
Sensor fault	Incorrect or out-of-range sensor value	Detect signal deviation
ECU fault	Incorrect controller response	Identify abnormal ECU behavior
CAN delay	Delayed CAN message	Detect timing deviation
Message loss	Missing network message	Flag communication failure
Network anomaly	Abnormal message frequency	Identify unusual network behavior
Connectivity fault	Cloud connection interruption	Detect connectivity failure

Table 1. Main fault categories used for validation

4.3 Comparative Verification

Two methods of verification have been analyzed by applying both to the same simulated system with the same faults. First, the traditional method of verification that is based on running predefined test cases according to the test sequences was used. Second, the novel AI-enabled framework with prioritization of test cases according to risk and system behavior and continuous analysis of anomalies was applied.

In case of any irregular reaction, the AI-integrated framework raises the priorities of associated test conditions and comes up with further scenarios around the area where failure has been found. This process of feedback helps to channelize the process of verification towards areas which may be weak in the system.

4.4 Performance Evaluation

The above two methods were evaluated based on test coverage, fault-detection ratio, verification time, high-risk situation coverage, and false-positive ratio. Test coverage means the portion of relevant situations of the system that was covered during the verification process, and fault detection ratio represents the number of faults found out of all the introduced faults. The verification time was used as an indicator of efficiency of the testing process, and the false-positive ratio represents normal situations considered abnormal.

The simulated validation provides a controlled basis for determining the benefits that the proposed model could provide. The metrics generated by this process will be used later on in this paper to measure the effectiveness of both the traditional methods of verification against those augmented by artificial intelligence.

5. Results and Discussion

The proposed AI-assisted verification methodology of the embedded system was tested using a simulated connected vehicle application, as mentioned above. In this experiment, the same system setup and fault instances as in a traditional verification process were used for comparison purposes. There were five criteria used to measure the effectiveness of the proposed solution: test coverage, fault detection percentage, verification time, critical scenarios test coverage, and false positives percentage.

Performance Parameter	Conventional Verification	AI-Assisted Verification
Test Coverage	78%	94%
Fault-Detection Rate	82%	96%
Verification Time (normalized units)	100	68
High-Risk Scenario Coverage	72%	95%
False-Positive Rate	9%	5%

Table 2. Comparison of Conventional and AI-Assisted Verification

The results indicate that the use of the AI-enabled approach led to increased test coverage as compared to the traditional one. Test coverage increased from 78% to 94%, representing a 16% increase. This increase was mainly due to intelligent test generation and selection, allowing verification to be conducted in more probable operating conditions. Traditional verification, on the other hand, relied on pre-set test cases, leading to lower coverage in operating conditions that were not so obvious.

Similar results were achieved in terms of fault detection capabilities. With the conventional verification technique, 82% of the faults were detected, while the system based on artificial intelligence managed to detect 96% of them. The role of the anomaly detection component here was played by the ongoing analysis of sensor discrepancies, ECU reactions, timing features, and communications. Thanks to that, the system managed to distinguish such problems as delayed CAN messages, incorrect sensor data, unusual ECU reactions, and connection interruptions.

There was a marked increase in the efficiency of the verification process. The verification time reduced from 100 units in conventional verification to 68 units using AI-aided verification, translating to an average of 32% savings in time. The efficiency in verification is due to the fact that the process focuses on high value test cases and prevents unnecessary repetition of low risk scenarios. The coverage of high risk scenarios also increased from 72% to 95%.

The false positive rate dropped from 9% to 5%. This is important due to the fact that a large number of incorrect fault detections can increase the time spent on reviewing by the engineers and damage the credibility of automatic verification systems. This result was made possible by using a combination of different behavioral criteria instead of one simple threshold.

All in all, the results of the simulation clearly show the possible benefits of using AI for improving the efficiency of verification. However, at the same time, these results were obtained in a controlled simulation and cannot be considered proof of equal capabilities of the approach in production cars. The effectiveness of the process would be contingent upon many different aspects including data quality, complexity of the system, chosen AI models, nature of the faults, etc. Therefore, further research with the use of actual ECU data and HIL testing is needed.

6. Conclusion and Future Work

An AI-enabled verification methodology for connected-vehicles is proposed to cope with increasing complexity in connected vehicle system verification. This methodology includes requirements engineering, test case generation using AI, connected-vehicle simulation, system monitoring, anomaly detection, verification analytics, and test optimization through a closed-loop verification process. This methodology is intended to help verification engineers to discover critical test cases, anomalies in the system, and reduce unnecessary verification processes.

The applicability of the suggested framework was assessed using the connected-vehicles simulation environment including ECUs, sensors, CAN communication, connectivity gateway, and vehicular-to-cloud communication. By introducing faults to the sensors, ECUs, communication channels, and connectivity, the comparison of the conventional verification process with AI-based verification process was made. Simulation results revealed that the suggested framework improves the test coverage from 78% to 94% and the detection rate of faults from 82% to 96%. Test coverage for risky scenarios was improved from 72% to 95%, while normalized verification time reduced by about 32%. False positive rate decreased from 9% to 5%. Overall, this demonstrates the potential of AI-based approaches to increase verification efficiency without sacrificing coverage of system conditions.

This current research is limited to simulation for validation purposes only; hence, the results of the framework need to be understood in terms of its feasibility and not performance of real vehicles. Further work should focus on deploying the framework in both Software-in-the-Loop and Hardware-in-the-Loop approaches, where actual data from automotive ECUs and networks can be used. Further research could consider the application of explainable AI techniques, adaptive fault injection, validation of V2X communications, over-the-air software updates, and cybersecurity-based validation. This would also lead to integration of digital twin technology and other existing automotive safety and security processes.

Funding: This research received no external funding.
Conflicts of Interest: The authors declare no conflict of interest.
Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers.

References

[1] X. Zhang *et al.*, "Vehicle-to-Everything Communication in Intelligent Connected Vehicles: A Survey and Taxonomy," *Automotive Innovation*, vol. 8, no. 1, 2025, doi: 10.1007/s42154-024-00310-2.

[2] Thota, V. N. K. (2026). AI-Integrated Structural Optimization Framework for Lightweight Heavy Fabrication Systems in Smart Manufacturing Environments. *Journal of Mechanical, Civil and Industrial Engineering*, *7*(3), 17-24. https://doi.org/10.32996/jmcie.2026.7.3.3

[3] C. Birchler, S. Khatiri, B. Bosshard, A. Gambi, and S. Panichella, "Machine learning-based test selection for simulation-based testing of self-driving cars software," *Empir. Softw. Eng.*, vol. 28, no. 3, 2023, doi: 10.1007/s10664-023-10286-y.

[4] X. Dong *et al.*, "Why Autonomous Vehicles Are Not Ready Yet: A Multi-Disciplinary Review of Problems, Attempted Solutions, and Future Directions," 2026. doi: 10.1002/rob.70108.

[5] W. Wang, K. Guo, W. Cao, H. Zhu, J. Nan, and L. Yu, "Review of Electrical and Electronic Architectures for Autonomous Vehicles: Topologies, Networking and Simulators," *Automotive Innovation*, vol. 7, no. 1, 2024, doi: 10.1007/s42154-023-00266-9.

[6] G. Payá-Vayá and H. Blume, *Towards a Common Software/Hardware Methodology for Future Advanced Driver Assistance Systems*. 2022. doi: 10.1201/9781003339847.

[7] S. Jana, Y. Tian, K. Pei, and B. Ray, "DeepTest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings - International Conference on Software Engineering*, 2018. doi: 10.1145/3180155.3180220.

[8] Thota, V. N. K. (2026). IoT-Enabled Cognitive Manufacturing Systems: A Conceptual Framework for Real-Time Autonomous Decision-Making in Industry 4.0. *Journal of Computer Science and Technology Studies*, *8*(5), 133-139. https://doi.org/10.32996/jcsts.2026.8.5.11

[9] M. N. Hossain, M. M. Rahman, and D. Ramasamy, "Artificial Intelligence-Driven Vehicle Fault Diagnosis to Revolutionize Automotive Maintenance: A Review," 2024. doi: 10.32604/cmes.2024.056022.

[10] Vasudevan Ananthakrishnan. (2026). Governance Frameworks for Large-Scale ETL Ecosystems in Complex Data Environments. *Journal of Computer Science and Technology Studies*, *8*(5), 82-87. https://doi.org/10.32996/jcsts.2026.8.5.5

[11] A. Shafique, A. Mehmood, and M. Elhadef, "Survey of Security Protocols and Vulnerabilities in Unmanned Aerial Vehicles," *IEEE Access*, vol. 9, 2021, doi: 10.1109/ACCESS.2021.3066778.

[12] R. Ben Abdessalem, A. Panichella, S. Nejati, L. C. Briand, and T. Stifter, "Testing autonomous cars for feature interaction failures using many-objective search," in *ASE 2018 - Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018. doi: 10.1145/3238147.3238192.

[13] Vasudevan Ananthakrishnan. (2026) Enterprise Data Migration Strategies for High-Assurance Information Systems. *Frontiers in Computer Science and Artificial Intelligence*. 5, 5 (Mar. 2026), 10–16. https://doi.org/10.32996/jcsts.2026.5.5.2