International Journal of Safety and Security Engineering
Vol. 15, No. 11, November, 2025, pp. 2191-2204
Journal homepage: http://iieta.org/journals/ijsse

Proximal Policy Optimization–Based Deep Reinforcement Learning for Intelligent Test Case Generation in Autonomous Vehicles

Tamizharasi Arthanari[1*], Rajalakshmi Dharmadurai[2], Keerthiga Viswanathan[3], Jaithunbi Abdul Kareem[4]

[1] Department of Computer Science and Engineering, R.M.D. Engineering College, Chennai 601206, India
[2] Department of Computer Science and Engineering, SRM Institute of Science and Technology, Ramapuram, Chennai 600089, India
[3] Department of Computer Science and Engineering, Anand Institute of Higher Technology, Kazhipattur 603103, India
[4] Department of Artificial Intelligence and Machine Learning, Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences (SIMATS), Chennai 602105, India

Corresponding Author Email: tamizh.cse@rmd.ac.in

https://doi.org/10.18280/ijsse.151101

Received: 4 August 2025
Revised: 5 October 2025
Accepted: 16 November 2025
Available online: 30 November 2025

Keywords:
Autonomous Vehicle, test case generation, Proximal Policy Optimization, Deep Reinforcement Learning, safety-critical scenarios, simulation-based testing, Markov Decision Process, fault injection

ABSTRACT

Autonomous Vehicles (AVs) require extensive and diverse testing to ensure safe and reliable operation under complex real-world driving conditions. Existing test case generation methods, including random sampling and rule-based scenario construction, cannot often adaptively expose rare and safety-critical events. This paper proposes a Proximal Policy Optimization (PPO)–based Deep Reinforcement Learning (DRL) framework for intelligent test case generation in autonomous driving systems. The problem is formulated as a Markov Decision Process (MDP), allowing a DRL agent to interact with the CARLA Simulation (CARLA) platform and iteratively synthesize challenging driving scenarios. The agent learns to adjust key parameters such as traffic density, vehicle behaviors, and environmental conditions to maximize the discovery of safety-critical events. PPO is adopted to ensure stable and sample-efficient policy learning during scenario generation. The framework is evaluated on thousands of simulated driving episodes across diverse urban and highway scenarios using metrics including safety-critical event detection rate, test coverage, and scenario diversity. Experimental results demonstrate over a 38% improvement in detecting safety-critical events compared with random testing and rule-based baselines, highlighting the effectiveness of the proposed system for improving AV validation and reliability.

1. INTRODUCTION

The growing need for quality of the software, speed of the delivery cycle, and the constant integration of the existing development environments has greatly increased the job of Quality Assurance (QA). The use of Agile and DevOps-driven processes has substituted conservative, siloed development patterns, necessitating QA processes to proceed continuously and adaptively as opposed to a detached and manual process [1]. Existing methods of QA, such as rule-based testing, manual testing, static testing, case definition, and the use of pre-existing datasets, are becoming less and less useful in testing and verifying large-scale complex software systems [2]. The mentioned limitations are particularly acute in the systems that may be described by real-time operation, cross-platform dependencies, micro service architecture, and dynamically changing operating conditions. This has led to the fact that the issue of reliability, efficiency, and safety before deployment is now a major concern when it comes to modern software engineering [3].

The existing QA methods are still extremely reliant on expert knowledge and manual intervention, which results in high development costs, limited scalability, incomplete test coverage, and exposure to human error. The creation of significant test data, especially edge cases that can expose latent system failures, is still considered a significant bottleneck [4]. Due to the increased complexity and autonomy of software systems, QA approaches have to change closer to the techniques of the past, including the non-static and manual approach in QA. Artificial Intelligence (AI) has become a disruptive technology in this field with the capacity to learn behavior from data, adapt to system behavior, and to engage in exploration of complex input space autonomously [5]. The AI-based QA systems are able to process application specifications, past defect data, and usage history to produce high-impact test cases, synthesize realistic and adversarial inputs, and rank tests by risk and probability of failure [6].

Such AI methods have been especially in the automated generation of test cases and test data. The process of aiding the translation of requirements or user stories into executable tests through Natural Language Processing (NLP) and exploring system behaviors through Reinforcement Learning (RL) agents can be autonomous and can be used to cover and identify execution paths that can lead to failure [7]. Machine

Learning (ML) models also facilitate prioritization of the tests by directing the validation towards the high-risk components. Although such methods have proved to be effective in the existing software systems, the implementation of the same to the Autonomous Vehicles (AVs) verification presents radically new issues that the existing QA models fail to address fully [8].

One of the safest and most complicated types of cyber-physical systems is autonomous driving systems. They are dependent on perception, decision-making, and control modules that are tightly coupled and interact in closed-loop with highly dynamic and uncertain environments. Although there are a lot of rapid developments in AV technologies, the standardized and scalable ways of validation have not developed yet [9]. Existing methods of testing, through manually specified scenarios, rule-based testing, or pre-defined test sets, cannot be used to represent such rare, interactive, and safety-critical situations in driving. This gap demonstrates that there is an urgent need to develop adaptive and data-driven methods of generating test cases that are specifically designed to test the AV safety requirements [10].

ML is one of the key components of AV systems, especially perception and decision-making modules. Object detection, semantic understanding, and end-to-end driving control have been popular applications of deep neural networks. These models are very high-dimensional and opaque, and it is very hard to describe their behaviour in all the available operating conditions [11]. Unexpected and unsafe actions can be observed because of minor changes in environmental conditions, traffic relations, or sensor inputs. Therefore, it is not possible to guarantee robustness and safety by means of exhaustive testing by existing methods [12].

Broader software testing studies have given black-box testing of Representational State Transfer (REST) APIs extensive interest because of comparable problems of large input spaces, state-dependent behaviour, and incomplete specifications. Numerous testing methods of REST APIs are based on the Open API Specification (OAS) to deduce good inputs and sequences of operation [13]. OAS tends to miss implicit logical relationships, business rules, and dynamic constraints, resulting in invalid test cases and poor coverage. Though the theory behind the REST API testing is quite different from that of AV validation, it can serve as a good analogy to the shortcomings of specification-driven testing with respect to limitations arising when system behaviour relies on hidden state and complicated interactions [14]. These problems are further compounded in AV systems by real-time dynamics, physical limitations, and safety demands, where even less effective testing with specifications enjoyed by a system is in effect [15].

The testing constraints in real life make the problem of validation even worse. Physical driving experiments to test the safety of AVs would take millions of kilometres and, to statistically confirm extremely autonomous driving functions, would take billions of kilometres, which is not feasible and would be economically prohibitive. Testing based on simulation has been adopted as a fundamental part of AV validation [16]. With scenario-based simulation, it is possible to systematically explore driving scenarios, and predefined scenarios themselves do not ensure adequate coverage of rare and critical events. This has prompted a rise in interest in the RL-based scenario generation, in which intelligent agents actively explore the scenario space to discover safety-critical behaviour [17].

The limitations of the standard testing strategies are also emphasized in complex driving conditions, e.g., intersection negotiation. The interaction of vehicles at intersections presents complex space and time relationships, and sophisticated arguments are necessary to prevent collisions and maintain the efficiency of the traffic [18]. Although there was a discussion on centralized and infrastructure-assisted traffic control methods, they are not as scalable and cannot be used in real-time. These difficulties also motivate the necessity of adaptive, learning-based testing structures that will be able to reveal failure situations that occur due to multi-agent interactions and dynamic decision-making [19].

Other methods that have been proposed to test the robustness of neural networks include adversarial testing and perturbation-based validation. Although useful at the component level, these methods can overlook dynamics at the system level, where unsafe behaviours can result due to the interplay between perception, planning, and control. System-level validation thus needs test methodologies that address behavioural results as opposed to a single model forecast [20].

Regardless of the major advancement, the existing AV test generation approaches are mostly stagnant, driven manually, or not adaptive enough. Learning based methods have been promising, and most of the RL algorithms, such as Deep Q-Networks (DQN) and A3C, are unstable and do not explore effectively in high-dimensional problems, especially in continuous control [21]. PPO is more stable and efficient in its sample usage is why it is suitable in safety-critical situations. Use in the generation of test cases of AVs and system-level safety validation is not well-investigated. To address this gap, this paper will concentrate on the PPO-based Deep Reinforcement Learning (DRL) to facilitate adaptive, efficient, and safety-based test case generation in AVs. Key contributions of the paper are as follows:

- Introduced a PPO-based framework for adaptive test case generation in AV simulations.
- Formulated scenario generation as a Markov Decision Process (MDP) to optimize failure-inducing test cases.
- Enhanced safety-critical scenario coverage by integrating DRL into the testing pipeline.
- Achieved a 38% improvement in fault detection rate over existing test generation methods.
- Validated framework performance using high-fidelity simulations targeting perception, planning, and control modules in AV systems.

2. RELATED WORKS

Test generation based on a rule and scenario library has been a popular technique in early AV-based validation. Such methods are based on manually specified traffic policies, professionally defined scenarios, or prescribed combinatorial sets of parameters to test the behavior of the system in known driving conditions. Although these approaches are interpretable and easily applicable, they have low scalability and subpar generalization [22]. Hand-operated scenario libraries are limited by human assumptions by nature and rarely represent the rare, compound, and emergent safety-critical scenarios caused by complex vehicle interaction. During the process of moving AV systems to open-ended and uncertain environments, rule-based testing has not been found to be adequate in unearthing unforeseen failure modes [23].

Search-based and optimization-based testing methods have

been suggested to enhance coverage of the scenario by performing a systematic search of the scenario parameter space. Other popular techniques are genetic algorithms, particle swarm optimization, and Bayesian optimization, which are applied to find the failure-causing inputs by maximizing the risk or violation measures specified by the user [24]. These techniques have proved to be effective in revealing the corner cases in the simulation environments. They usually, however, depend on handwritten fitness functions, handwritten representation of a static scenario, and offline optimization, which hinder their dynamism in dealing with changes in system behaviour. Search-based algorithms typically have difficulty in scaling to high-dimensional continuous spaces, and can tend to prematurely underscore local optimality [25].

Adversarial scenario generation and robustness testing are concerned with identifying vulnerabilities in perception and decision-making modules by adding small but specific perturbations to sensor inputs or environmental parameters. Adversarial attacks using gradients and falsification methods have been used to detect unsafe behaviours in neural network constituents and closed-loop control systems [26]. Although these methods yield important information on the sensitivity of models, they are generally component-based or a partial observer of a system. Consequently, they can miss the failure modes due to long-term interactions, temporal dependencies, or multi-agent dynamics that appear in the driving situation in the real world [27].

The test case generation based on RL has received growing popularity because of its sequential decision-making process representation of scenario generation. Challenging traffic behaviors, adversarial agents, and environment settings that are maximally likely to cause collisions or safety violations have been generated by RL agents. Value-based algorithms (DQN or actor-critic (A3C) algorithms were the most used in early studies [28]. Despite potential advantages, such methods are typically unstable to training, inefficient to explore, and sensitive to reward design, especially in high-dimensional and continuous simulation tasks such as autonomous driving. Such constraints restrict their use to scalable and reliable AV safety validation [29].

In contrast to existing RL-based testing systems, the proposed one uses PPO to attain stable and sample-efficient as well as scalable scenario generation. In contrast to acting on top of value or asynchronous action critic, such as PPO, policy updates are limited to avoid the destructive policy update, leading to robust training in challenging driving conditions. The given framework is specifically focused on system-level safety-critical event discovery, where scenario parameters are dynamically adjusted to observed closed-loop behaviors as opposed to fixed risk measures. This can be used to better explore rare and high-impact cases, better test coverage, and better fault detection than existing RL-based and optimization-driven testing strategies. Therefore, the PPO framework is an important breakthrough in adaptable, learning-based AV test case generation.

3. PROBLEM FORMATION

Test case generation for AVs is a complex task that aims to expose failures in the perception, planning, and control modules under diverse and challenging driving scenarios. The goal is to automate this process using an RL agent that interacts with a simulated driving environment to generate scenarios that maximize the likelihood of system failure or sub-optimal behavior. This can be modeled as an MDP defined by the tuple:

$$M = (S, A, P, R, \gamma) \tag{1}$$

where, S: Set of states representing environment and vehicle conditions (e.g., weather, traffic, road layout). A: Set of actions (eg, modify speed, insert pedestrian, change visibility). $P(s'|s,a)$: Transition probability from state s to s' after action a. $R(s,a)$: Reward function that quantifies test case effectiveness (e.g., system failure, safety violations). $\gamma \in [0,1]$: Discount factor that balances immediate vs. future rewards. The objective is to learn a policy $\pi_\theta = (a|s)$ that maximizes the expected cumulative reward:

$$J(\theta) = E_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)\right] \tag{2}$$

To optimize this policy, the PPO algorithm is used. PPO updates the policy by solving:

$$L^{CLIP}(\theta) = E_t\big[min\big(r_t(\theta)\hat{A}_t, clip(r_t(\theta)1 - \epsilon_1 + \epsilon)\hat{A}_t\big)\big] \tag{3}$$

where, $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the probability ratio between new and old policies. $\hat{A}_t$ is the advantage estimate at time step t. ϵ is the clipping parameter to prevent large updates. The reward function $R = (s,a)$ is designed to encourage generation of challenging test scenarios:

$$R(s,a) = \lambda_1 Risk(s,a) + \lambda_2 Coverage(s) - \lambda_3 Similarity(s,T) \tag{4}$$

where, $Risk(s,a)$: Likelihood of causing failure or unsafe behavior. $Coverage(s)$: Contribution to overall scenario diversity. $Similarity(s,T)$: Redundancy with respect to proposed test suite T. $\lambda_1, \lambda_2, \lambda_3$: Weighting coefficients.

4. MATERIALS AND METHODS

The proposed system integrates a DRL agent trained via PPO within a simulation-based autonomous driving environment, such as CARLA Simulation (CARLA) or LGSVL. The simulator models real-world conditions, including traffic, pedestrians, road layouts, weather, and vehicle dynamics. The autonomous system's behavior control decisions, path planning, and safety violations are continuously monitored. The problem is formulated as an MDP, where the agent modifies scenario variables (e.g., adding obstacles or altering visibility) and receives rewards based on system responses shown in Figure 1. PPO updates the policy mapping states to scenario modifications, aiming to identify failure-prone or safety-critical events. The reward function emphasizes scenario diversity, novelty, and risk (e.g., near-miss detection). A fully connected A3C framework with a CNN processes sensor data like camera and LiDAR inputs. Scenario effectiveness is evaluated by comparing fault detection rates, system resilience, and diversity against baseline methods such as random scenario generation and rule-based approaches.

To develop a robust and reliable mandatory lane change method, this study proposes an RL approach based on PPO. The following sections will describe the system architecture, state space, action space, and reward components of the proposed decision-making framework. Figure 1 illustrates the overall system structure designed to support autonomous lane changes. The system comprises two main components: A simulation environment and a learner architecture.

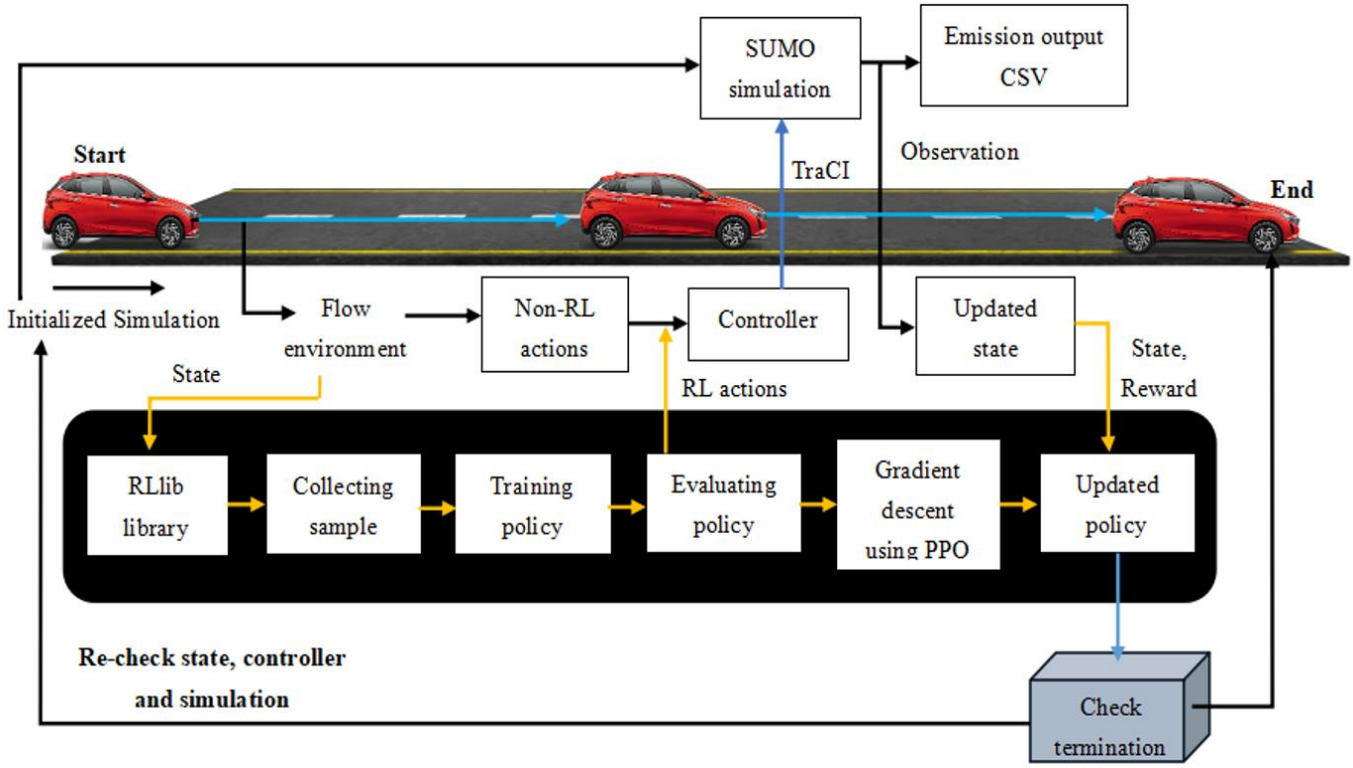


Figure 1. Proposed architecture

The learner component enables high-level decision-making, accesses vehicle data embedded in the road infrastructure, and takes into account vehicle dynamics generated within the simulation environment.

4.1 Dataset description

The dataset used in this study was generated using a high-fidelity autonomous driving simulator like CARLA or LGSVL, designed to replicate real-world scenarios shown in Table 1. It includes features across behavioral, vehicular, environmental, and infrastructure dimensions. Environmental conditions (e.g., rain, fog, and lighting), road types, and dynamic elements like pedestrians or debris simulate high-risk situations. Sensor inputs such as LiDAR and RGB images support perception tasks, while real-time vehicle data tracks performance. Binary flags indicate events like collisions or lane departures. Each scenario is tagged with a unique ID and outcome label, enabling robust training and evaluation of the PPO-based DRL test scenario generator.

4.2 State and action space

The lane-change decision-making process considers the states of five vehicles, consisting of the ego vehicle and four surrounding vehicles. These vehicles jointly influence the Autonomous Vehicle's decision to keep its lane or execute a lane change. The state space captures the kinematic and positional information required for safe and efficient maneuvering, while the action space includes discrete lateral and longitudinal decisions that jointly define the agent's control policy. A summary table of state and action space is shown in Table 2.

Table 1. Dataset description for simulation-based test case generation

Feature Category	Description
Environmental Conditions	Clear, Rainy, Foggy, Cloudy, Night, Day
Road Infrastructure	Urban, highway, rural Number of lanes (1 to 6)
Traffic Elements	Number of vehicles per km Number of pedestrians per block or meter
Dynamic Obstacles	Vehicle, animal, debris, bicycle Static moving, erratic
Vehicle State	AV speed in km/h 30 coordinates of the vehicle in the simulation map
Sensor Data	Front-facing camera image 30 spatial perception data
AV System Performance	Indicates if a collision occurred Indicates if the vehicle left its designated lane
Test Case Metadata	Unique identifier for each generated scenario Indicates if the scenario led to AV failure or suboptimal behavior

4.3 State and action space for lane change decision in Autonomous Vehicles

The autonomous driving scenario consists of the following

vehicles:

Ego Vehicle: C_e

Surrounding Vehicles:

- C_0: Leading vehicle in the existing lane
- C_1: Leading vehicle in the target lane
- C_2: Following vehicle in the existing lane
- C_4: Following vehicle in the target lane

This notation is used consistently throughout the manuscript and matches the simulation implementation.

State Space S

S is composed of 21 continuous variables, derived as follows.

Ego Vehicle State

$$S_{C_e} = \{i_e, v_e, a_e, y_e, v_e^{lat}\} \tag{5}$$

where, i_e: longitudinal position; v_e: longitudinal speed; a_e: longitudinal acceleration; y_e: lateral position; v_e^{lat}: lateral speed

Surrounding Vehicle State S_{C_x} (4 variables each)

For each $C_x \in \{C_0, C_1, C_2, C_3\}$

$$S_{C_x} = \{d_{rel}^x, v_x, a_x, y_x\} \tag{6}$$

where, d_{rel}^x: relative distance to ego vehicle; v_x: longitudinal speed; a_x: acceleration; y_x: lateral position. Thus, the total state space is:

$$S = S_{C_e} + \sum_{x=0}^{3} S_{C_e} = 5 + (4 \times 4) = 21\ continuous\ variables \tag{7}$$

Action Space (A): The action space 1 includes lateral and longitudinal commands for the agent:

Lateral Actions $A_{lat} \in \{0, 1, 2\}$: 0: Lane keeping; 1: Initiate lane change; 2: Abort lane change maneuver.

Longitudinal Actions $A_{long} \in \{0, 1\}$: 0: Follow existing lane leader; 1: Follow target lane leader

The combined high-level action space is:

$$A = A_{lat} \times A_{long} = \{(0,0), (0,1), (1,0), (1,1), (2,0), (2,1)\} \tag{8}$$

Total of 6 discrete action combinations.

Table 2. Summary of state and action space components for lane change decision

Component	Variables	Description
Ego Vehicle	5	Position, speed, acceleration, lateral motion
Each Surrounding Vehicle	4 × 4	Relative distance, speed, acceleration, and lateral position
Total State Dimension	21	Continuous
Lateral Actions	3	Keep, change, abort
Longitudinal Actions	2	Follow the existing or target leader
Total Actions	6	Discrete

4.4 Test case generation objective (with Proximal Policy Optimization)

To generate optimal test cases using PPO, the goal is to find a policy $\pi_\theta(a|s)$ that maximizes the expected cumulative reward: $J(\theta) = E_{\pi_\theta}[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)]$ (9)

where, $s_t \in S$: state at time t; $a_t \in A$: action taken at time t; $R(s_t, a_t)$: reward function; γ: discount factor. The reward function is designed to encourage safe and efficient lane changing while avoiding collisions:

$$R(s_t, a_t) = \lambda_1 r_{safe}(s_t, a_t) + \lambda_2 r_{smooth}(s_t, a_t) - \lambda_3 r_{risk}(s_t, a_t) \tag{10}$$

where, r_{safe}: reward for maintaining safety distance; r_{smooth}: reward for smooth transitions (IDM-based); r_{risk}: penaity for collision or abrupt maneuvers; λ_1, λ_2, λ_3: weighting coefficients. The low-level controller uses a modified Intelligent Driver Model (IDM) to execute the longitudinal behavior smoothly based on high-level PPO decisions.

Where d is negative while the car is departing and positive when it is moving toward the center. In Figure 2, the state formulation is displayed. Everyone employs every vehicle's speed for the activity. Ultimately, an 8-dimensional action space and a 16-dimensional state space are built.

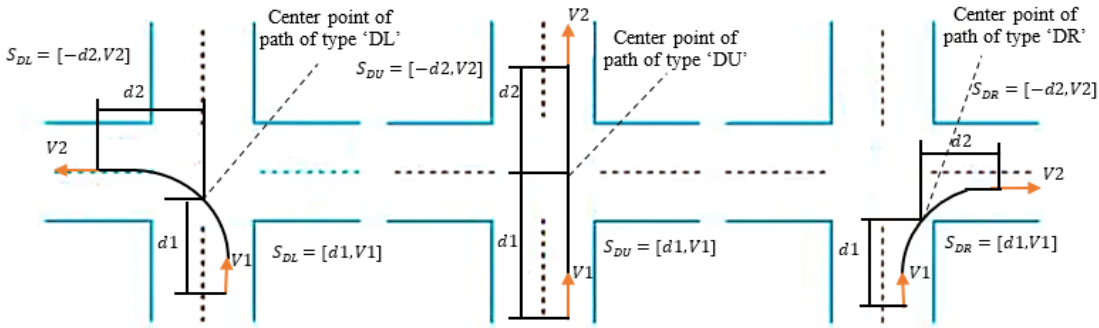


Figure 2. State formulation

4.5 Reward function

The reward function in the proposed PPO-based test case generation framework is carefully crafted to balance comfort, efficiency, and safety-the three core objectives in designing an intelligent and human-like lane change behaviour. The total reward at time step t is defined as:

$$R(s_t, a_t) = \lambda_1 r_{comfort}(s_t, a_t) + \lambda_2 r_{efficiency}(s_t, a_t) - \lambda_3 r_{safety}(s_t, a_t) \tag{11}$$

where, λ_1, λ_2, $\lambda_3 \in R^+$: are the weights assigned to each component. $r_{comfort}, r_{efficiency}, r_{safety}$: normalized scalar values representing rewards/penalties.

Comfort Evaluation $r_{comfort}$: Minimizes jerk (rate of change of acceleration) to ensure smooth driving experience:

$$r_{comfort} = -(\alpha|j_{lat}| + \beta|j_{long}|) \quad (12)$$

where, $j_{lat} = \frac{da_{lat}}{dt}$, lateral jerk; $j_{long} = \frac{da_{long}}{dt}$, longitudinal jerk; α, β: comfort sensitivity factors.

Efficiency Evaluation $r_{\text{efficiency}}$: Encourages timely and goal-oriented lane changes with minimal deviation from the target lane:

$$r_{\text{efficiency}} = -(\eta_1 t_{travel} + \eta_2 d_{rel}^{target}) \quad (13)$$

Where, t_{travel}: time taken to complete the lane change; d_{rel}^{target}: relative distance to the center of the target lane; η_1, η_2: efficiency weighting parameters.

Safety Evaluation r_{safety}: Penalizes risky behaviour such as collisions or near miss events with surrounding vehicles:

$$r_{\text{safety}} = \begin{cases} -R_1 & if\ collision\ occurs \\ -R_2 & if\ near\ collision\ (headway\ below\ threshold) \\ 0 & otherwise \end{cases} \quad (14)$$

where, R_1, R_2 safety penalty constants. Near collision if time-to-collision (TTC) < Threshold (e.g., 2.5 sec).

The incentive function is used to assess every evaluation scenario in order to ascertain whether the PPO agent learns to optimize convenience, effectiveness, and safety shown in Table 3. Low-jerk, prompt, and collision-free lane changing behaviors are encouraged by the reinforcement learnt strategy.

Table 3. Test case definitions based on the reward function

Test Case ID	Scenario Description	Expected Risk	Comfort Violation	Efficiency Deviation
TC_LC_001	Lane change in foggy conditions with high-speed oncoming C_3	High	Low	Moderate
TC_LC_002	Aborted lane change due to merging vehicle in blind spot	Medium	Medium	High
TC_LC_003	Smooth lane change in low traffic with no interference	Low	Low	Low
TC_LC_004	Late lane change on a highway exit ramp	High	High	High
TC_LC_005	Lane change in dense traffic with slow reaction time	High	Medium	Medium

4.6 Proximal Policy Optimization

The PPO method is used as the baseline in this study due to its ease of implementation and consistent policy improvement. Although PPO is based on Trust Region Policy Optimization (TRPO), it is more efficient because of two key innovations: Generalized Advantage Estimation (GAE) and an unconstrained surrogate objective function.

4.6.1 Unconstrained surrogate objective function

To achieve stable and consistent policy improvement, TRPO limits policy updates to avoid significant deviations. PPO streamlines this approach by using a clipped surrogate objective function, which penalizes updates that stray too far from the existing policy, eliminating the need for complex constraints. The PPO objective function is defined as:

$$J_{PPO}(\theta) = E_{s,a}\left[min\left(\rho_t(\theta)\hat{A}_t, clip(\rho_t(\theta)1 - \epsilon_1 + \epsilon)\hat{A}_t\right)\right] \quad (15)$$

where, $\rho_t(\theta) = \frac{\pi_\theta(s|a)}{\pi_{\theta_{old}}(s|a)}$ the probability ratio between the new and old policies. $\hat{A}_t$: The estimated advantage at time step t. ϵ is a small hyperparameter (e.g., 0.2) that restricts the update range.

4.6.2 Generalized advantage estimation

The advantage function is critical for calculating policy gradients in PPO. It is estimated using the difference between the action-value function and the state-value function:

$$\hat{A}^\pi(s_t, a_t) = \hat{Q}^\pi(s_t, a_t) - V(s_t, w) \quad (16)$$

where, $\hat{Q}^\pi(s_t, a_t)$ is the estimated action-value function. $V(s_t, w)$ is the estimated value of state s_t with parameters w.

Monte Carlo Estimate (used in TRPO):

$$\hat{Q}^\pi(s_t, a_t) = \sum_{l=t}^{\infty} \gamma^{l-t} r_l \quad (17)$$

This estimate is unbiased but exhibits high variance.

One-Step Bootstrapping (used in A3C):

$$\hat{Q}^\pi(s_t, a_t) = r_t + \gamma V(s_{t+1}, w) \quad (18)$$

This method has lower variance but introduces bias. To strike a balance between bias and variance, GAE uses a weighted sum of multi-step temporal difference errors:

$$(s_t, a_t) = \sum_{l=t}^{\infty} \gamma\lambda^{l-t}\, \delta_l \quad (19)$$

where the temporal difference (TD) error δ_l defined as:

$$\delta_l = r_l + \gamma V(s_{l+1}, w) - V(s_l, w) \quad (20)$$

Here, $\lambda \in [0,1]$ a hyperparameter that governs the bias-variance trade-off. A lower λ introduces more bias with less variance, while a higher λ does the opposite.

4.7 Intelligent test case generation

The challenge of identifying program flaws through execution on carefully selected input data is a central focus in testing. A program P is considered to contain an error if its output does not match the expected requirements or is deemed incorrect by the tester. Verifying output correctness and

generating relevant test scenarios requires some knowledge of the correct program, which may or may not be sufficiently complete. Ideally, program testing involves using input/output samples to demonstrate or probabilistically validate the program's equivalence to a correct version or executable specification. Testing DRL agents remains in its early stages. Similar to DL agents trained through supervised learning, DRL agents are also susceptible to adversarial attacks that generate conflicting scenarios. The approach for DRL differs significantly, as it emphasizes configuring the environment for each scenario encountered by the agent, rather than altering raw sensor inputs. The sampling strategy used as a baseline in the existing study closely aligns with this approach. Empirical findings demonstrate that the search-based method performs better than others in inducing a broader and more diverse range of failures across multiple case studies. A method has been proposed to assess the adaptability of DRL agents by reinitializing learning in environments different from those encountered during the initial training phase. This enables the construction of an adaptation frontier distinguishing between scenarios in which the agent adapts successfully and those in which it fails.

Although other methodologies also generate new environmental configurations, the existing approach does not retrain the agent. Instead, the focus is on testing to identify vulnerabilities in the agent during evaluation. Explored training and testing DRL agents in procedurally generated environments. Specifically, their study utilized a set of algorithmically generated 3D mazes to train agents, followed by a local search process that modified the mazes during evaluation based on agent performance. This technique produced out-of-distribution configurations not represented in the original training dataset. In contrast to directly testing agent performance in these altered environments, the existing study employs a failure predictor, a proxy representation of the environment is to reduce the computational cost of the search.

In more complex environments than mazes, it becomes prohibitively expensive to execute the DRL agent within the environment at every search iteration. To address evaluation challenges, a recent search-based strategy was proposed to assess the quality of DRL agents. Their approach involves sampling the environment to identify a reference trace that successfully solves the RL task. This trace is constructed using a depth-first search algorithm and consists of all states not pruned during the search's backtracking process.

The key aspects being measured include fault detection effectiveness, scenario diversity, safety-critical coverage, and comfort in vehicle trajectories. The null hypothesis (H_0) assumes that the PPO-generated test cases are no better or inferior in performance compared to baseline methods (such as random or rule-based test generation).

This is expressed mathematically as:

$$H_0: \mu_{PPO} \leq \mu_{baseline}$$

where, μ represents the average effectiveness score, which is a weighted sum of fault detection rate F, scenario diversity D, and safety event coverage C. These weights w_1, w_2, w_3 are chosen based on the Importance of each factor to the test generation goals.

In contrast, the alternative hypothesis (H_1) asserts that the PPO-based framework outperforms the baseline, aiming to generate more useful, diverse, and fault-revealing test cases:

$$H_1: \mu_{PPO} \leq \mu_{baseline}$$

To deepen the analysis, sub-hypotheses are introduced:

- $H_1.1$ checks whether PPO-generated cases trigger more faults, indicating stronger stress-testing capability.
- $H_1.2$ evaluates behavioral diversity using variance across features like traffic density, weather, and obstacle types.
- $H_1.3$ measures coverage of safety-critical scenarios, ensuring the policy does not miss important corner cases.
- $H_1.4$ focuses on trajectory comfort, using mean jerk as a proxy for smoothness in the vehicle's movement.

Each equation ties directly to a measurable property of the system, enabling objective evaluation using statistical tests (e.g., t-test or Wilcoxon test) over generated test data. The reward function in the PPO model encourages policies that optimize these objectives, and testing these hypotheses validates that the optimization successfully aligns with real-world testing needs.

4.8 Proximal Policy Optimization-driven Deep Reinforcement Learning framework for intelligent test case generation in Autonomous Vehicles

Objective: Maximize the expected cumulative reward for generating test cases that challenge AV's decision-making under safety, efficiency, and comfort constraints.

Input: Environment model $\mathcal{E}$; Initial policy network $\pi_\theta(a|s)$; Value network $V(s; w)$; Hyperparameters: Discount factory γ, GAE factor λ, Clipping threshold ϵ, batch size B, epochs U

Output: Optimized policy π_θ for generating effective test cases

Initialization: 1. Randomly initialize policy parameters θ and value function parameters w; Initialize replay buffer $B \leftarrow \emptyset$

Step 1. Collect Environment Rollouts

Generate episodes using the existing policy π_θ and collect a batch mathcal:

$$D = \{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^{B} \tag{21}$$

Step 2: Estimate the Temporal Difference (TD) Error

For each step t, compute:

$$\delta_t = r_t + \gamma V(s_{t+1}, w) - V(s_t, w) \tag{22}$$

Step 3: Generalized Advantage Estimation (GAE)

$$\hat{A}_t = \sum_{t=0}^{\infty} (\gamma\lambda)^l \delta_{t+1} \tag{23}$$

This balances bias and variance in the estimation of the advantage function.

Step 4: Value Function Target

$$\hat{V}_t = \hat{A}_t + V(s_t, w) \tag{24}$$

Step 5: Policy Update with PPO Surrogate Objective

Define the probability ratio:

$$\rho_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{25}$$

Clip the objective to avoid large policy updates:

$$J_{PPO}(\theta) = E_{s,a}\big[min\big(\rho_t(\theta)\hat{A}_t, clip(\rho_t(\theta)1-\epsilon_1 + \epsilon)\hat{A}_t\big)\big] \tag{26}$$

Update policy network:

$$\theta \leftarrow \theta + \alpha_\theta \nabla_\theta J_{PPO}(\theta) \tag{27}$$

Step 6: Value Network Update: Minimize squared error between predicted and target value:

$$J_v(w) = \frac{1}{B}\sum_{t=1}^{B}\big(V(s_t,w) - \hat{V}_t\big)^2 \tag{28}$$

Update value network:

$$w \leftarrow w - \alpha_v \nabla_w J_V(w) \tag{29}$$

Step 7: Repeat Steps 1-6: Repeat for multiple iterations until convergence or maximum episodes reached.

Reward Function Design: The reward $R(s_t, a_t)$ balances three objectives:

$$R(s_t, a_t) = \lambda_1 r_{comfort} + \lambda_2 r_{efficiency} + \lambda_3 r_{safety} \tag{30}$$

With the help of this method, the PPO agent may learn a lane-change decision policy that emphasizes realism, variety, and edge-case exposures while producing difficult yet secure test scenarios for self-driving vehicles.

4.9 Test case generation

The testing process begins with analysis, identifying key factors—such as ego vehicle acceleration—that influence driver assistance technologies. Based on these criteria, relevant test scenarios are developed. The third phase involves executing these scenarios in a Software-in-the-Loop (SiL) simulation environment. In the evaluation phase, scenario criticality is assessed using metrics like time-to-collision, which serve as reward signals for the DRL agent to guide scenario refinement. The exploration phase follows, leveraging prior knowledge or probing new environmental conditions. Parameter adjustment is performed using a ϵ-greedy algorithm that balances optimal and random actions to improve scenario diversity. Each action corresponds to modifying a parameter, prompting the generation of a new test instance, and restarting the cycle. Finally, the save critical test cases phase retains high-impact scenarios for future analysis.

Step 1: Analysis: This initial step identifies the key influencing factors on the automated driving function under investigation, these may include: Speed of the ego vehicle v_e; Relative positions and velocities of surrounding vehicles $\{v_x, d_x\}$; Road curvature ρ, weather *W*, and traffic density *T*. These parameters form the input feature vector $i \in R^n$ which defines the scenario space.

Step 2: Test Case Generation: Using the parameters identified in Step 1, initial test scenarios are generated. The test cases are defined as a vector:

$$P = \{v_e, v_0, d_0, \rho, W, T, \dots\} \tag{31}$$

These parameters are input into a simulation environment. Initially, parameter values can be chosen randomly or based on predefined distributions, as the RL agent will iteratively refine them.

Step 3: Test Run Execution: Each test case from Step 2 is executed using a SiL simulation framework. The simulation simulates vehicle dynamics, sensor models, and the autonomous driving software stack. During this phase, key runtime metrics are collected, such as: Time-to-Collision TTC; Lane deviation; Braking and acceleration behavior.

Step 4: Test Evaluation: The criticality of each scenario is evaluated using metrics like Time-to-Collision (TTC):

$$TTC = \frac{d_{rel}}{v_{rel}} \; where \; v_{rel} = v_e - v_{obstacle} \tag{32}$$

A reward function is constructed to encourage the generation of dangerous but realistic cases:

$$R(s_t, a_t) = -\alpha TTC^{-1} + \beta comfort\ penalty \tag{33}$$

Scenarios with low TTC values are rewarded more as they are closer to critical conditions.

Step 5: Exploration: Using the PPO RL agent, new test scenarios are explored. To balance exploration and exploitation, an epsilon-greedy strategy is used:

$$a = \begin{cases} random\ (A), & with\ probability\ \epsilon \\ \arg max_a\, Q(s,a), & with\ probability\ 1-\epsilon \end{cases} \tag{34}$$

where, $Q(s,a)$: Estimated action-value; A: Action space, i.e, parameter modifications; $\epsilon \in [0, 1]$: Exploration rate (typically decaying over time).

Step 6: Parameter Change: Based on exploration, test parameters are adjusted. An action here refers to increasing or decreasing a parameter:

$$P_x' = P_x + \Delta P, \quad where\ \Delta P \in \{-\delta, 0, +\delta\} \tag{35}$$

This modifies the scenario to potentially make it more critical. For example: Increasing vehicle speed v_e; Decreasing following distance d_0; Changing road curvature *p*. The newly generated parameter set becomes the next test case.

Step 7: Save Critical Test Cases: Scenarios that meet or exceed a criticality threshold (eg, TTC < 2s) are stored for future safety validation and regression testing. A critical test case repository ensures: Reusability in future training and validation. Coverage of edge conditions and failure-prone cases. Save if: $R(s_t, a_t) > R_{threshold}$.

The cycle repeats by feeding back the modified parameters into Step 2, forming a closed-loop system for intelligent test case generation. The RL agent continuously adapts its strategy to find edge cases that challenge the autonomous driving stack, while ensuring diversity, realism, and safety-critical exposure.

5. SIMULATION SETUP

To evaluate and generate critical test scenarios, a simulation system replicates realistic highway driving using SiL architecture. It combines a physics-based vehicle dynamics

model with a perception and control stack typical of AVs. The scenario involves five cars: the ego vehicle and four surrounding vehicles in existing and target lanes. A high-level decision module, trained with PPO, governs the ego vehicle's behavior, while low-level lane-change heuristics and a modified Intelligent Driver Model (IDM) execute actions shown in Figure 3. The state space includes 21 features capturing kinematics and relative positions, while the action space defines discrete longitudinal and lateral maneuvers. Multiple episodes are simulated with varied traffic, road curvature, and weather to ensure diversity. Key metrics like time-to-collision and trajectory smoothness are logged and used to refine the PPO policy. Simulations are reproducible, using fixed random seeds and standardized inputs to ensure consistency in evaluation.

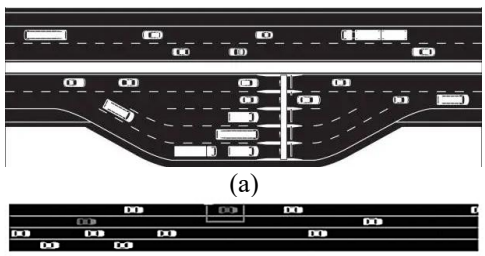
To guarantee consistent and effective learning, the PPO framework for AVs test case creation makes use of precisely calibrated hyper parameters shown in Table 4. Future reward relevance and learning variance are balanced by a GAE value of 0.95 and a discount factor of 0.99. Safe policy changes are ensured by a clipping threshold of 0.2. The policy and value learning rates are set at 3×10^{-4} and $1 \times 10^{-3,}$ respectively. Ten training periods and a batch of 2048 steps are used in each PPO update. Exploring is made possible via a ε-greedy approach with ε = 0.1. These environments encourage the creation of a variety of crucial situations for assessing and testing self-driving cars. The durations of MA-PPO and PPO-DRL episodes first rise quickly before leveling off is shown in Figure 4. This may be explained by the fact that the learnt strategy initially primarily concentrates on avoiding collisions, as doing so would result in a significant negative reward. The lengthy episode duration results from the cars' tendency to wait or travel extremely slowly until there is no chance of an accident.

Figure 3. Simulation network

With this configuration, agents can be deployed in various traffic conditions using different PPO- DRL techniques, state representations, and reward structures. A vehicle scope with $\Lambda_{ahead} = 2$ and $\Lambda_{behind} = 1$ was used in the studies. This configuration enhances the agent's anticipation capability and ensures continuous visibility of all lanes on a three-lane roadway. The unsignalized junction had a range of 200–220 meters. Although numerous scenarios are typically simulated in the field, the existing study focused specifically on the performance of the leading AVs at an unsignalized intersection. When approaching the intersection, platooning vehicles proceeded straight in four distinct directions. AV penetration rates were evaluated in 10% increments, ranging from 1% to 100%. All vehicles making left turns or changing lanes at the incomplete intersection were excluded from consideration. The scenario involving the leading AVs at the non-signalized junction is illustrated in Figure 5. To highlight the effectiveness of the leading AV scenario, comparisons were made with other configurations, including a leading human-driven vehicle scenario and an all-human-driven vehicle scenario. The comparison of these experiments at the non-signalized intersection is presented in Figure 6.

Table 4. Hyper parameter settings for PPO-based test case generation

Parameter	Symbol	Value
Discount Factor	γ	0.99
GAE Parameter	λ	0.95
Clipping Threshold	ϵ	0.2
Learning Rate (Policy)	α_θ	$3e^{-4}$
Learning Rate (Value)	α_w	$1e^{-3}$
Number of Epochs per Update	U	10
Batch Size	B	2048
Mini-batch Size	-	64
Total Timesteps	-	1,000,000
Exploration Rate	ε	0.1
PPO Update Frequency	-	Every 2048 steps
Max Episode Length	-	1000 steps



(a)

(b)

Figure 4. Illustration of driving scenarios: (a) Highway driving; (b) Highway merging

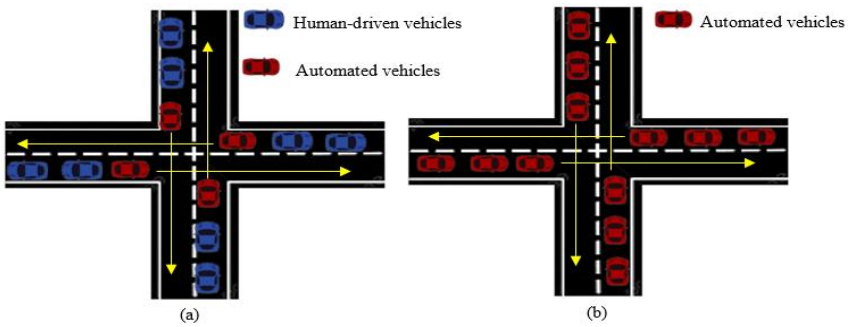


Figure 5. Autonomous Vehicle (AV) experiments at an unsignalized intersection: (a) Mixed-autonomy traffic with AV penetration rates varying from 10% to 90% in 10% increments; (b) Fully autonomous traffic with 100% AV penetration

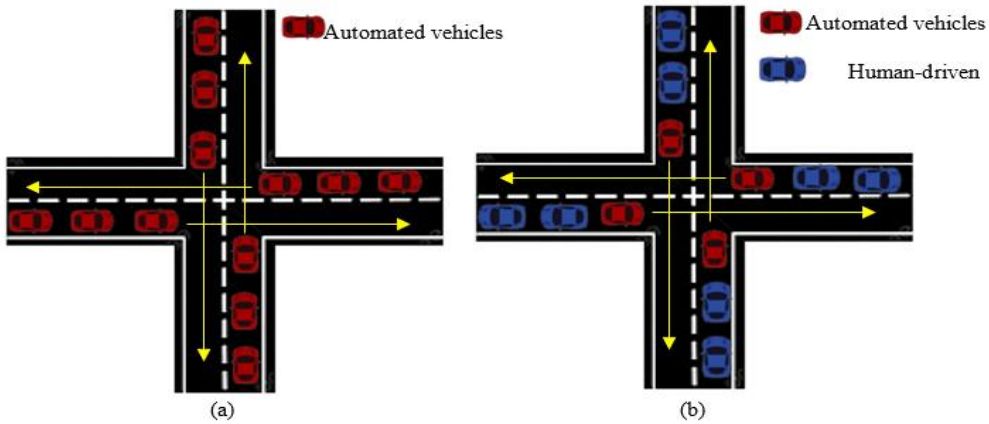
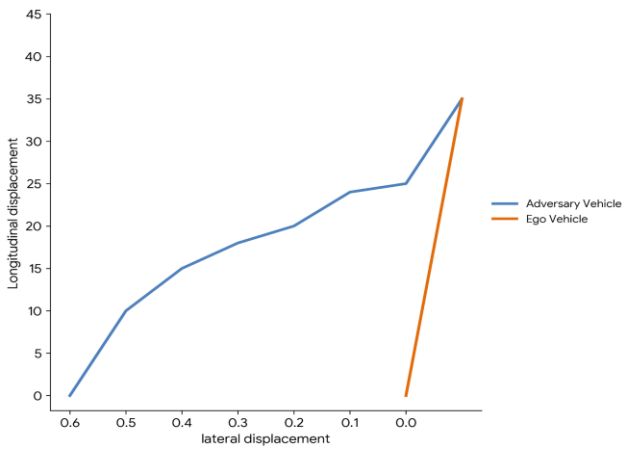


Figure 6. Experimental comparison at an unsignalized intersection: (a) Scenario with 100% human-driven vehicles (0% AV penetration); (b) Scenario with a leading human-driven vehicle and varying Autonomous Vehicle (AV) penetration rates from 10% to 90% in 10% increment

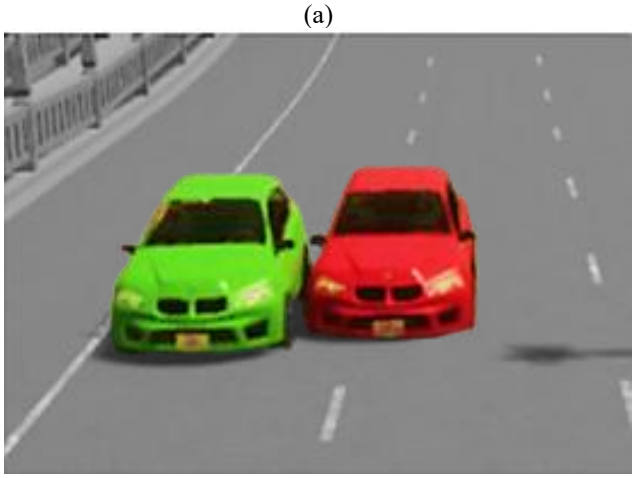
Random samples from each univariate distribution were initially combined to generate a trial scenario. As shown in Figures 7(a) and (b), the opponent vehicle initiated a lane change ahead of the ego vehicle and collided with its rear, though this interaction did not reflect real-world data. In the second experiment, a multivariate normal distribution fitted via Kernel Density Estimation (KDE) captured inter-variable relationships. Figures 7(c) and (d) show a more realistic side-impact crash, where the opponent vehicle, just behind the ego vehicle at time step 21, collided during a lane change. Figure 7(e) demonstrates that the PPO-DRL method generated realistic trajectories. A cut-in collision was detected when the opponent vehicle underestimated the merge gap. As seen in Figure 7(f), the opponent changed lanes at time step 4 and impacted the ego vehicle by time step 21.

The proposed PPO-DRL-based test case generation framework demonstrates superior performance compared to four existing approaches shown in Figure 8. It achieves an accuracy of 96.8%, indicating strong classification between critical and non-critical scenarios. With a recall of 97.2%, it effectively detects most true critical cases, while its 95.4% precision confirms minimal false alarms. The F1-score of 96.3% highlights a balanced performance in both precision and recall. These results emphasize the effectiveness of integrating RL with PPO to generate diverse, realistic, and high-risk scenarios for AVs testing.

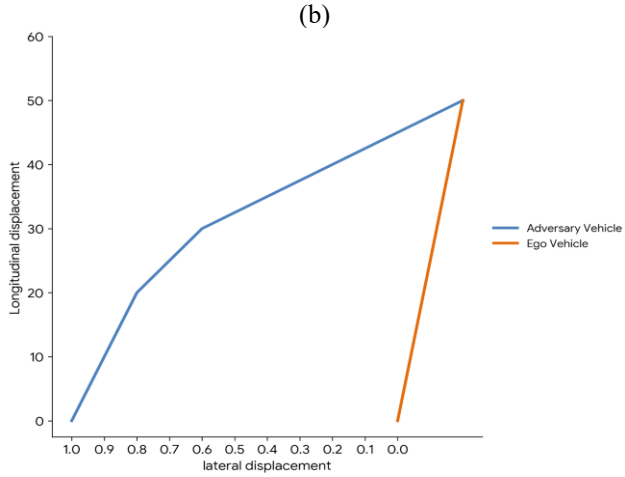
(a)



(b)

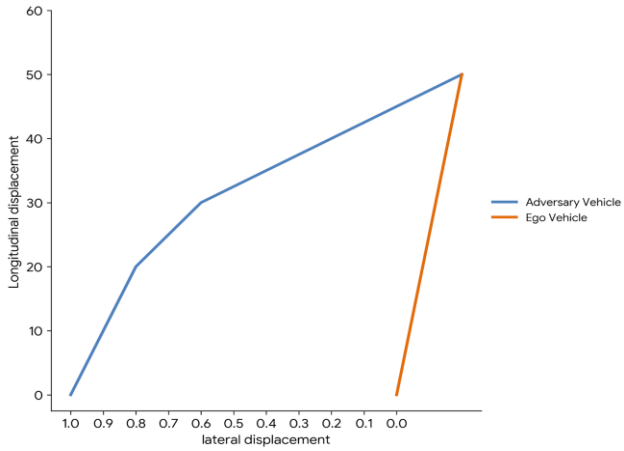


(c)



(d)

(e)



(f)

Figure 7. Testing Autonomous Vehicle (AV) and ego vehicles using proposed system based on 3 scenarios

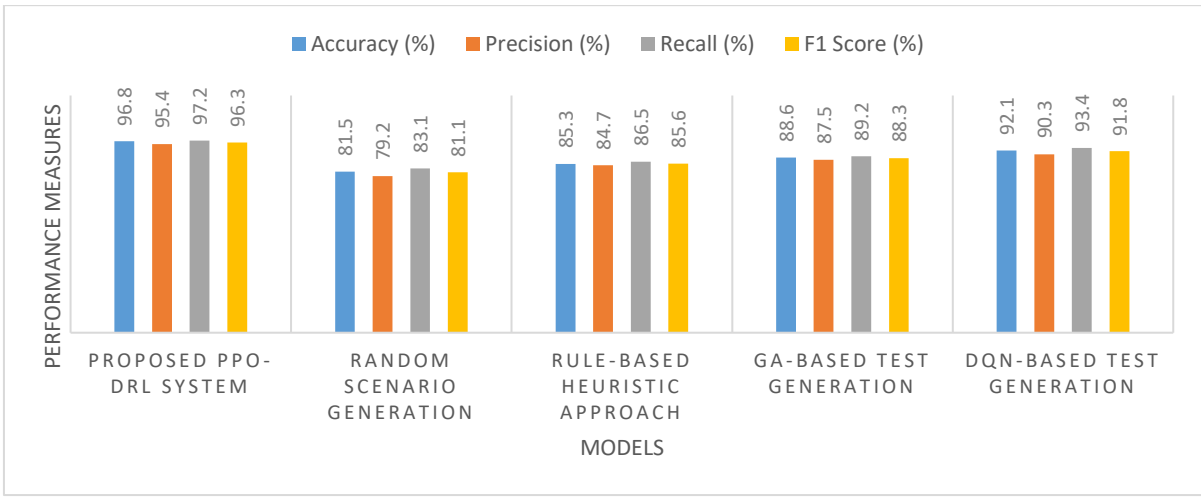


Figure 8. Performance measures

Table 5. Performance measures (time to collision, coverage, scenario diversity, execution time)

System	Time-to-Collision (s)	Coverage (%)	Scenario Diversity (Entropy)	Execution Time (s/case)
Proposed PPO-DRL System	1.4	94.2	0.89	1.2
Random Scenario Generation	3.6	62.3	0.48	0.6
Rule-Based Heuristic Approach	2.9	70.5	0.52	0.8
GA-Based Test Generation	2.1	81.6	0.65	2.0
DQN-Based Test Generation	1.8	88.7	0.74	1.6

The proposed PPO-DRL-based test case generation framework demonstrates strong performance across key metrics shown in Table 5. It achieves the lowest average Time-to-Collision (TTC) of 1.4 seconds, highlighting its ability to generate highly critical scenarios. With maximum scenario coverage (94.2%) and high scenario variety (entropy = 0.89), it effectively explores diverse driving situations.

The proposed PPO-DRL system achieves the highest critical case rate of 41.8%, demonstrating its strong ability to generate high-risk scenarios that effectively challenge AV systems shown in Table 6. It also records the highest cumulative reward score of +37.6, reflecting an optimal balance among efficiency, comfort, and safety-criticality. In contrast, random scenario generation consistently underperforms, with the lowest critical case rate (12.4%) and a negative reward score (–15.3). While DQN offers moderate improvement, it still falls short of PPO's performance.

Table 6. Comparison of the critical case ratio and reward score of proposed and existing systems

System	Critical Case Rate (%)	Reward Score (R)
Proposed PPO-DRL System	41.8	+37.6
Random Scenario Generation	12.4	–15.3
Rule-Based Heuristic Approach	18.6	+4.8
GA-Based Test Generation	26.9	+18.2
DQN-Based Test Generation	33.7	+27.4

The proposed PPO-DRL architecture demonstrates excellent adaptability, achieving a maximum training accuracy of 98.1% and testing accuracy of 96.8% shown in Figure 9. This indicates strong generalization and effective learning of critical scenario features. In contrast, random scenario generation yields the lowest accuracy, reflecting poor learning capability. While rule-based and GA-based methods show moderate improvements, they exhibit noticeable gaps between training and testing performance. The DQN-based model performs better in consistency and precision but reaches only 92.1% testing accuracy. These results confirm the PPO framework's effectiveness as a robust, intelligent approach for generating high-quality test scenarios in AV environments. The proposed PPO-DRL-based system achieves the lowest testing loss (0.022) and training loss (0.014), demonstrating superior adaptability and learning efficiency shown in Figure 10. These low values indicate consistent performance across unseen scenarios and effective modelling of complex relationships needed for generating critical test cases. While the DQN-based model performs better than random generation, it still incurs higher losses than PPO-DRL. Rule-based and GA-based methods show moderate results. Overall, the PPO-DRL framework proves effective in learning optimal test generation strategies while minimizing overfitting.

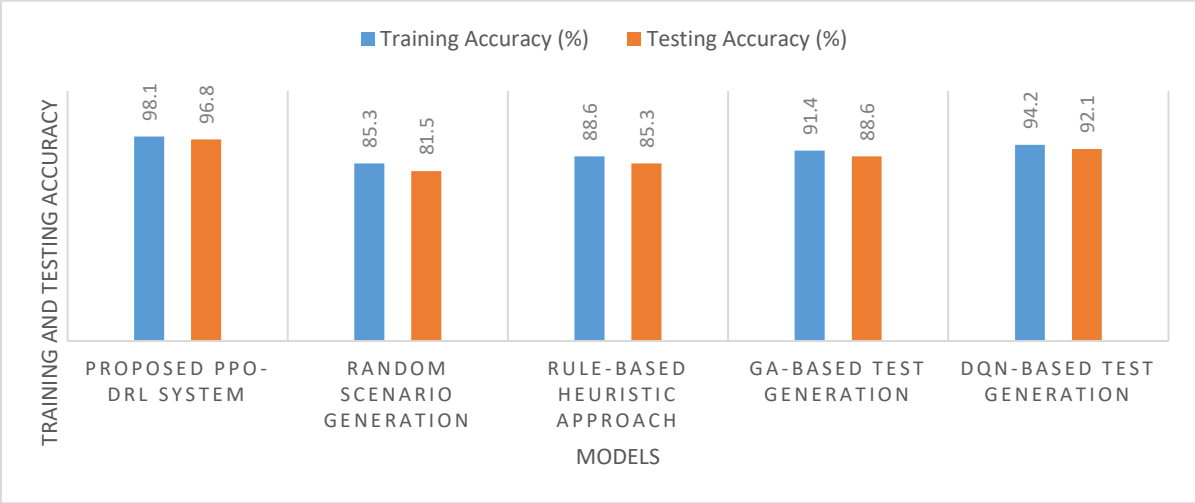


Figure 9. Comparison of training and testing accuracy

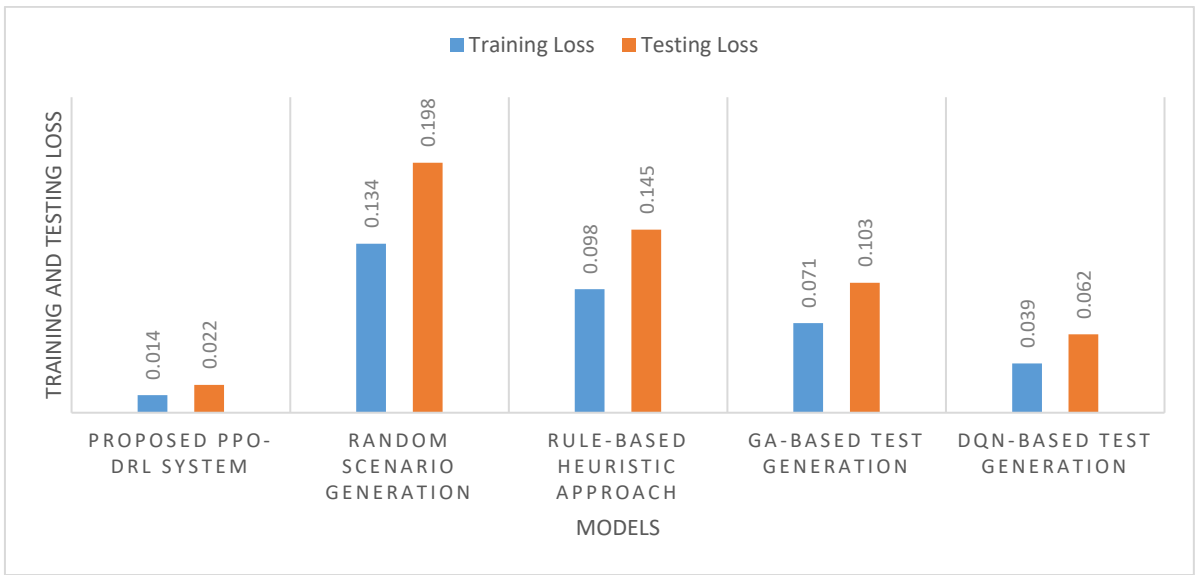


Figure 10. Comparison of training and testing loss

6. CONCLUSIONS

The implementation of the PPO-DRL framework for intelligent test case generation demonstrates substantial progress in AV safety validation. The proposed approach consistently outperforms existing methods across multiple evaluation metrics, achieving a testing accuracy of 96.8%, an F1-score of 96.3%, and a low testing loss of 0.022, indicating strong generalization capability. Moreover, the framework effectively uncovers safety-critical scenarios, reflected by a high critical case rate of 41.8%, broad scenario coverage of 94.2%, and high diversity (entropy of 0.89). The reduced time-to-collision (1.4 s) further highlights its ability to expose challenging edge cases that stress AV decision-making systems. Stable policy optimization is ensured through PPO's clipped objective and generalized advantage estimation, enabling reliable learning and refinement. Despite these strengths, the existing study is limited by its reliance on a

single simulation environment and its focus on specific highway and unsignalized intersection scenarios, without direct real-world validation. Future work will address these limitations by extending the framework to multi-agent traffic settings, incorporating additional driving scenarios, and integrating hardware-in-the-loop or real-world testing to further enhance robustness and practical applicability.

REFERENCES

[1] Rehman, U. (2025). Proximal Policy Optimization-driven decentralized peer-to-peer energy trading model for optimal real-time operations in smart energy communities. Energy Efficiency, 18: 49. https://doi.org/10.1007/s12053-025-10330-4

[2] Li, T.T., Li, S.Q., Ding, C.X., Bao, Z., Alhazmi, M. (2025). Intelligent wireless power scheduling for lunar multienergy systems: DRL for real-time adaptive beam steering and vehicle-to-grid energy optimization. International Transactions on Electrical Energy Systems, 2025(1): 9877968. https://doi.org/10.1155/etep/9877968

[3] Zhang, X.H., Bai, W.Q., Liu, J., Yang, S.N., Shang, T., Liu, H.L. (2025). Enhancing geomagnetic navigation with PPO-LSTM: Robust navigation utilizing observed geomagnetic field data. Sensors, 25(12): 3699. https://doi.org/10.3390/s25123699

[4] Liu, H.Q., Li, T., Jiang, F.Y., Su, W.K., Wang, Z.C. (2024). Coverage optimization for large-scale mobile networks with digital twin and multi-agent RL. IEEE Transactions on Wireless Communications, 23(12): 18316-18330. ttps://doi.org/10.1109/TWC.2024.3464639

[5] Tong, H., Chu, L., Wang, Z.X., Zhao, D. (2025). Adaptive Pulse-and-Glide for synergistic optimization of driving behavior and energy management in hybrid powertrain. Energy, 330: 136622. https://doi.org/10.1016/j.energy.2025.136622

[6] Huang, B., Yu, W., Ma, M., Wei, X., Wang, G. (2025). Artificial-Intelligence-based energy management strategies for hybrid electric vehicles: A comprehensive review. Energies, 18(14), 3600. https://doi.org/10.1016/j.energy.2025.136622

[7] Long, Y.S., Gong, S.M., Sun, S.M., Lee, G.C.F., Li, L.H., Niyato, D. (2025). Lyapunov-guided DRL for semantic-aware Aoi minimization in UAV-assisted wireless networks. IEEE Transactions on Wireless Communications, 24(8): 6351-6364. https://doi.org/10.1109/TWC.2025.3552809

[8] Gao, Y., Piccinini, M., Zhang, Y.C., Wang, D.R., et al. (2025). Foundation models in autonomous driving: A survey on scenario generation and scenario analysis. arXiv preprint arXiv:2506.11526. https://doi.org/10.48550/arXiv.2506.11526

[9] Wang, X., Zhou, J., Feng, Y.L., Mei, J.H., Chen, J.M., Li, S. (2025). Dashing for the golden snitch: Multi-drone time-optimal motion planning with multi-agent RL. In 2025 IEEE International Conference on Robotics and Automation (ICRA), Atlanta, GA, USA, pp. 16692-16698.https://doi.org/10.1109/ICRA55743.2025.11128442

[10] Zhang, H., Yang, G.X., Lei, N., Chen, C.Y., Chen, B.L., Qiu, L. (2025). Scenario-aware electric vehicle energy control with enhanced vehicle-to-grid capability: A multi-task RL approach. Energy, 335: 138189. https://doi.org/10.1016/j.energy.2025.138189

[11] Zhong, L.L., Liu, Y., Wang, L.Q., Zhao, J., Wang, W. (2024). Interval type-2 fuzzy DRL-based operational optimization of industrial aerodynamic system. IEEE Transactions on Instrumentation and Measurement, 73: 1-13. https://doi.org/10.1109/TIM.2024.3413155

[12] Wahid, A., Mirza, M.A., Ahmed, M., Sheraz, M., Chuah, T.C., Lee, I.E., Khan, W.U. (2024). Towards secure and scalable vehicular edge computing with zero-energy RIS using DRL. IEEE Access, 12: 129330-129346. https://doi.org/10.1109/ACCESS.2024.3457853

[13] Raguvaran, S., Anandamurugan, S. (2024). Enhancement of energy utilization efficiency and speed control of autonomous electric vehicles (AEVs): A hybrid approach. Energy Efficiency, 17: 59. https://doi.org/10.1007/s12053-024-10238-5

[14] Chen, Z.L., Pan, S.G., Yu, K.G., Wu, Y.T., Gao, W., Wang, Z.X. (2025). Fusion control tracking strategy for autonomous vehicles: A fast PPO RL based on attention mechanism and physical information. IEEE Transactions on Intelligent Transportation Systems, 26(11): 18906-18920. https://doi.org/10.1109/TITS.2025.3609483

[15] Raeisi, M., Sesay, A.B. (2024). Power control of 5G-connected vehicular network using PPO-based DRL algorithm. IEEE Access, 12: 96387-96403. https://doi.org/10.1109/ACCESS.2024.3427124

[16] Hu, B., Jiang, L., Zhang, S., Wang, Q. (2023). An eplainable and robust motion planning and control approach for autonomous vehicle on-ramping merging task using DRL. IEEE Transactions on Transportation Electrification, 10(3), 6488-6496. 10.1109/TTE.2023.3347278

[17] Wu, Q., Ji, M.X., Fan, P.Y., Wang, K.Z., Cheng, N., Chen, W., Letaief, K.B. (2025). PPO-based vehicle control for ramp merging scheme assisted by enhanced C-V2X. arXiv preprint arXiv:2501.12656. https://doi.org/10.48550/arXiv.2501.12656

[18] Taheri, H., Hosseini, S.R., Nekoui, M.A. (2024). DRL with enhanced PPO for safe mobile robot navigation. arXiv preprint arXiv:2405.16266. https://doi.org/10.48550/arXiv.2405.16266

[19] Siboo, S., Bhattacharyya, A., Raj, R.N., Ashwin, S.H. (2023). An empirical study of DDPG and PPO-based RL algorithms for autonomous driving. IEEE Access, 11: 125094-125108. 10.1109/ACCESS.2023.3330665

[20] Sharma, R., Garg, P. (2024, October). RL advances in autonomous driving: A detailed examination of DQN and PPO. In 2024 Global Conference on Communications and Information Technologies (GCCIT), BANGALORE, India, pp. 1-5. 10.1109/GCCIT63234.2024.10862532

[21] El-Hariry, M., Richard, A., Muralidharan, V., Geist, M., Olivares-Mendez, M. (2024). DRIFT: DRL for intelligent floating platforms trajectories. In 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Abu Dhabi, United Arab Emirates, pp. 14034-14041. http://doi.org/10.1109/IROS58592.2024.10801927

[22] Ma, D.F., Chen, X., Ma, W.B., Zheng, H.R., Qu, F.Z. (2023). Neural network model-based RL control for AUV 3-D path following. IEEE Transactions on Intelligent Vehicles, 9(1): 893-904. http://doi.org/10.1109/TIV.2023.3282681

[23] Yang, Z.W., Tang, J.M., Cai, L. (2025). Multi-scenario automatic parking based on DRL. In The Proceedings of the 11th International Conference on Traffic and Transportation Studies. ICTTS 2024. Lecture Notes in Civil Engineering, pp. 481-488. https://doi.org/10.1007/978-981-97-9644-1_52

[24] Bingol, M.C. (2025). A safe navigation algorithm for differential-drive mobile robots by using fuzzy logic reward function-based DRL. Electronics, 14(8): 1593. https://doi.org/10.3390/electronics14081593

[25] Shen, J., Zheng, F.H., Chen, T.Y., Deng, W., Bellotti, A., Tesema, F.B., Lucchi, E. (2025). Optimizing urban land-use through DRL: A case study in hangzhou for reducing carbon emissions. Land, 14(12): 2368. https://doi.org/10.3390/land14122368

[26] Wang, C.Q., Wang, Y. (2024). Safe autonomous driving with latent dynamics and state-wise constraints. Sensors, 24(10): 3139. https://doi.org/10.3390/s24103139

[27] Gutiérrez-Moreno, R., Barea, R., López-Guillén, E., Arango, F., Abdeselam, N., Bergasa, L.M. (2023). Hybrid decision making for autonomous driving in complex urban scenarios. In 2023 IEEE Intelligent Vehicles Symposium (IV), Anchorage, AK, USA, pp. 1-7. https://doi.org/10.1109/IV55152.2023.10186666

[28] Fan, J.C., Lei, X.Y., Chang, X.L., Miši, J., Miši, V.B., Yao, Y.Y. (2025). Less is more: A stealthy and efficient adversarial attack method for DRL-based autonomous driving policies. IEEE Internet of Things Journal, 12(15): 30215-30227. https://doi.org/10.1109/JIOT.2025.3569877

[29] Li, H.Q., Wang, Y.Z., Pan, M.Y., Li, S.X., Guan, W. (2025). DRL based joint resource allocation and service migration for smart-buoy-enabled maritime multi-access edge computing networks. IEEE Internet of Things Journal, 12(24): 52687-52705. https://doi.org/10.1109/JIOT.2025.3612960