Autonomous Vehicle Design Conformity Validation in Simulation Using Reinforcement Learning

Mohammed Eleffendi and M. Ilhan Akbas
Embry-Riddle Aeronautical University
Daytona Beach, Florida
eleffenm@my.erau.edu, akbasm@erau.edu

ABSTRACT

The validation of cyber-physical systems (CPS) is a critical challenge, particularly in safety-critical domains such as autonomous vehicles (AVs) and robotic control systems. Traditional validation techniques often struggle to efficiently explore the vast state and action spaces of these systems, limiting their ability to uncover failure scenarios and assess robustness under diverse conditions. Recent advancements in Reinforcement Learning (RL) have introduced new methodologies for system validation, leveraging RL's ability to discover optimal or near-optimal policies while systematically covering a wide range of possible system states. This capability makes RL particularly useful for stress-testing CPS algorithms by exposing them to rare or extreme conditions that traditional testing approaches might overlook. In this paper, we present an RL-based validation approach designed to evaluate the robustness and compliance of a cyber-physical system's motion planning algorithm in simulation. The framework is applied to an adaptive cruise control (ACC) system in an AV, demonstrating its effectiveness in identifying failure cases where the system deviates from its specified design requirements. Compared to conventional testing approaches, our method provides a more systematic and automated means of discovering failure-inducing scenarios, ensuring broader coverage of the operational space. The results illustrate how RL-driven validation can complement traditional CPS verification techniques by efficiently identifying edge cases and potential safety violations. While this study focuses on autonomous vehicle applications, the proposed RL-based framework has the potential to extend to a wide range of CPS domains, including robotics, industrial automation, and aerospace systems. By integrating RL into the validation pipeline, this research contributes to the development of more resilient, compliant, and robust CPS architectures capable of operating reliably in complex, dynamic environments.

ABOUT THE AUTHORS

Mohammed Eleffendi is a Software Engineer at Avidyne. He received his Master of Science in Electrical and Computer Engineering from the Embry-Riddle Aeronautical University and his Bachelor of Science in Mechatronics, Robotics, and Automation Engineering from the University of Jordan. He has research interests in artificial intelligence, space systems and flight operating systems.

M. Ilhan Akbas is an associate professor at the Electrical Engineering and Computer Science Department, director of the Flexible & Intelligent Complex Systems (FICS) research group, co-director of WIDE Lab, and a member of the leadership at the Center for Aerospace Resilient Systems (CARS) at Embry-Riddle Aeronautical University (ERAU). He has over 15 years of experience in machine learning, complex systems, modeling and simulation. He received his Ph.D in Computer Engineering at the University of Central Florida (UCF), where he conducted research at the Artificial Intelligence (AI) Things Laboratory. His background also includes eight years of defense industry and enterprise level software development experience. His research has secured funding through grants from agencies such as National Science of Foundation, Federal Aviation Administration, US Air Force, Office of Naval Research, as well as industry, which also resulted in more than 30 journal and 80 conference publications in IEEE, ACM and AIAA venues. He is a senior member of IEEE, and a member of ACM, AIAA, SAE, International Alliance for Mobility Testing and Standardization, Complex Systems Society and Cyber Safety Commercial Aviation Team.

Autonomous Vehicle Design Conformity Validation in Simulation Using Reinforcement Learning

Mohammed Eleffendi and M. Ilhan Akbas
Embry-Riddle Aeronautical University
Daytona Beach, Florida
eleffenm@my.erau.edu, akbasm@erau.edu

INTRODUCTION

Reinforcement Learning (RL) is a subfield of Machine Learning (ML) that focuses on teaching agents how to make a sequence of decisions by interacting with an environment (Akinci et al., 2024). RL has been gaining a lot of attention due to its success in certain problem areas. One of the earlier events that drew attention to RL was when DeepMind successfully trained an agent that outperforms humans on the board game Go. The use and popularity of RL has been increasing ever since and different problem domains have been making use of RL for different tasks such as job scheduling on high performance computers (Liang et al., 2020), supporting intelligent healthcare systems (Abdellatif et al., 2023), providing cybersecurity (Fard et al., 2023), optimizing chemical reactions (Zhou et al., 2017) and automated driving (Sallab et al., 2017; Summer et al., 2025).

The power of RL approaches comes from their ability to learn the actions (policy) that will maximize the total reward over the long run (Barto and Sutton, 1998) without a model of the system. For example, consider the task of falsifying an Adaptive Cruise Control (ACC). A well-designed RL falsifier learns an acceleration policy, if such policy exists, that will cause the ACC to crash the lead vehicle on which the RL falsifier resides and cause it to fail to achieve the basic requirement of avoiding a crash. The first step to solve a problem using RL is to formulate it using a Markov Decision Process in which the agent starts within the environment at state S_t, at time step t=0, takes an action a_t and goes from state S_t to state S_(t+1) and receives a reward R_(t+1). If the reward was good, then the agent updates itself in a way that makes it choose action a_t more often. If the reward was bad, then the agent updates itself in a way that makes it choose actions that produce a better reward than action a_t should such actions exist. The agent continues to interact with the environment until an optimal or near-optimal policy is learned. The RL problem structure is shown in Figure 1.

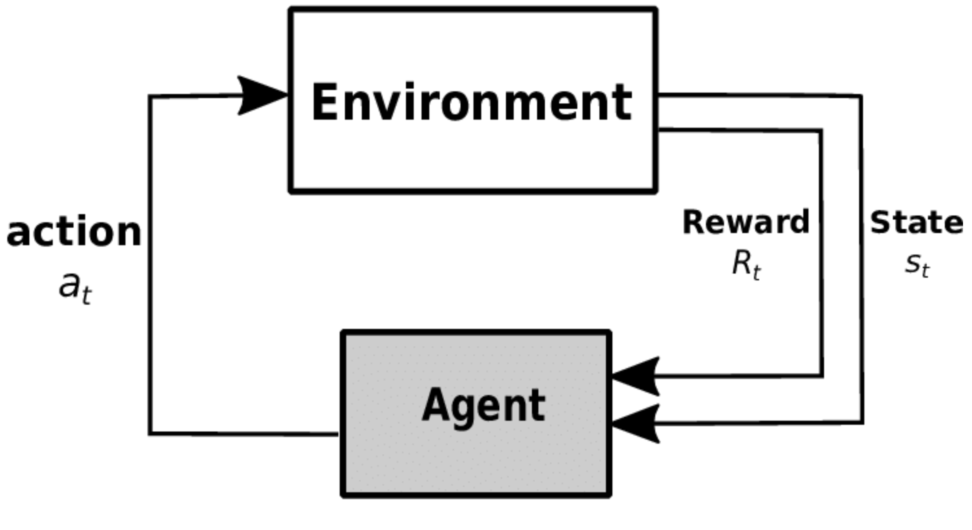

Figure 1. Reinforcement learning problem structure.

Some challenges in training an RL agent are crafting a good reward function and, in case of off-line training, having an accurate model of the environment (Barto and Sutton, 1998). It is critical to balance exploration, trying new actions to discover their potential rewards, and exploitation, choosing the best-known actions to maximize reward. In most cases, the reward function needs to be tuned until the agent behavior is satisfactory since poorly designed rewards can lead to unintended behavior or inefficiencies. Hence, both the design and crafting of RL approaches are very important for the correct system behavior.

In this paper, we propose a framework for testing the robustness and conformity of a cyber-physical system to its design requirements. All engineering systems are designed according to specified sets of design requirements. Hence, it is critical to test these systems according to their requirements. This is also a vexing challenge for the pervasive deployment of autonomous vehicles (AVs) (Medrano-Berumen et al., 2020). Our approach leverages RL to assess the robustness and adherence to design specifications of an autonomous vehicle motion planning algorithm. A methodology with two RL agents is proposed for this purpose and evaluated on an ACC system. The results of the experiments show that the approach succeeds in identifying input trajectories that cause the system to fail to behave according to its design requirements. This information is invaluable to tune system parameters, improve the system design or define the validity region of the system.

The remainder of this paper is organized into Related Work, Methodology, Performance Evaluation, and Conclusion sections. The Related Work section outlines the current literature for the use of RL in autonomous vehicles and testing related research, summarizing their unique approaches. The Methodology describes the components of our approach to the problem of testing the conformity of a cyber-physical system to its requirements. The Performance Evaluation section describes how the methods were evaluated and what the expected outcome is for each experiment. This section also presents the results of the experiments and discusses how the methodology performed. Finally, the Conclusion section summarizes the content of the paper, ending with the broader impacts of the research and the prospects of future work on this subject.

RELATED WORK

The validation of cyber-physical systems, particularly autonomous systems, is a critical topic with a vast interest (Razdan et al., 2023) (Akbas, 2021). Some of the existing approaches include the application of RL. These studies propose methods or frameworks that make use of RL for cyber-physical systems falsification or testing. While most of them focus on autonomous or assisted driving systems, some present more generalized methods for scenario generation in cyber-physical systems testing, which make them similar to our framework and approach.

One of the important use cases of RL has been to develop innovative falsification and adversarial testing approaches for autonomous systems. Yamagata et al. proposed a method for falsification of cyber-physical systems using deep RL (Yamagata et al., 2020). Their method is based on using an RL agent to find an input trajectory to the system that minimizes its robustness which is measured by a set of properties formulated using Signal Temporal Logic. The agent is rewarded for trajectories to which the system is least robust. Those properties can be thought of as design requirements. Our framework in this paper does not depend on formulating those properties in Signal Temporal Logic. Instead, the designer/tester has the freedom to craft his own reward function. Doreste et al. used RL to model the agents in the safety validation scenarios as independent autonomous agents (Doreste et al., 2024). These agents challenge the vehicle under test to guide it towards a collision while the training mainly focuses on the improvement of the vehicle's robustness against these challenging behaviors. Cai et al. also presented an adversarial testing approach using RL and focused on the stress testing (Cai et al., 2024). They integrate Responsibility Sensitive Safety and Dynamic Time Warping theories to develop a reward function that will use the evolving direction of the agents in the scenarios to explore vulnerabilities of the vehicle under test. DeepCollision addresses the challenge of an ever-changing environment by employing RL agents that engage dynamically with their surroundings (Lu et al., 2022). It utilizes Deep Q-Learning to discover environmental setups that cause AVs to collide, using the likelihood of a collision as the criterion for the reward function. Feng et al. (Feng et al., 2023) use a similar approach for improving the intelligence of the testing environment to increase the efficiency of the testing procedure. They use RL for training the vehicles in the scenario to learn when to execute the adversarial maneuvers that will cause collisions with the vehicle under test.

There are safety testing frameworks in the literature that use RL as a core component. Behzadan et al. (Behzadan and Munir, 2019) proposed a framework for benchmarking collision avoidance mechanisms in AVs. Their benchmarking framework involves training an RL agent to manipulate the trajectory of the AV such that it collides with any vehicle or object other than the adversary, or shift its natural trajectory towards an arbitrary alternative. This framework is suited for comparing the robustness of different collision avoidance mechanisms. Qin et al. (Qin et al., 2023) also proposed a testing framework tailored for cyber-physical systems, with a specific focus on operations in uncertain environments. These applications require defining the environment to include all possible realistic operational scenarios that the system might face. The testing process seeks to pinpoint operational scenarios where the evaluated system fails to meet its defined criteria. The authors utilize deep RL across three example systems in autonomous driving, executed within a photo-realistic autonomous driving simulator. Koren et al. (Koren et al., 2018) proposed a framework that focuses on the adaptive stress testing of AV. Their framework involves training an RL agent to perturb stochastic elements in the environment of an AV until it is involved in a crash. Specifically, they trained an RL agent to induce collision for a vehicle that employs the Intelligent Driver Model (Treiber et al., 2000) by manipulating the amount of noise going into the vehicle's sensors and moving a pedestrian in front of it. (Wang et al., 2020) proposed an approach for improving the generalization of trained policies to safety-critical scenarios. They use an adversarial RL agent to apply disturbances to the training environment of another RL agent so that the latter generalizes better.

The discussed methodologies collectively facilitate the progression of techniques for developing safety validation scenarios for AVs, utilizing diverse theoretical frameworks and testing methodologies to improve the validation processes of autonomous driving systems. While the majority of existing studies emphasize theoretical methodologies aimed at generating challenging scenarios, in this paper we aim to address the disconnect between testing and design requirements.

METHODOLOGY

We propose a framework to test the conformity of a cyber-physical system to its design requirements by encoding those requirements in the reward function of an RL agent whose function is to treat the system as a black box and test it by applying perturbations to its environment. The difference between our framework and other frameworks in literature is that our framework can be applied to domains and not only AVs. For instance, it can be applied to test a drone autopilot design requirements such as robustness, maximum overshoot or response time when subjected to wind gusts or other disturbances. Another example would be testing a legged-robot's design requirements to environmental variations or disturbances.

Adaptive Cruise Control and Reinforcement Learning

We use an Adaptive Cruise Control (ACC) system to implement and demonstrate our framework. An ACC adjusts the speed of the ego vehicle such that it maintains a safe distance to the lead vehicle. The falsifier aims to succeed in identifying a set of initial conditions and acceleration commands that cause the ego vehicle ACC to fail its most basic requirement of avoiding a crash. This information can be used by the ACC designer to identify weaknesses in the system and tune a more reliable and robust ACC.

The ACC algorithm from the "Adaptive Cruise Control System Using Model Predictive Control" (Mathworks, 2024) is used in our approach. This ACC algorithm operates as follows: If the relative distance is greater than or equal to the safe distance, the ACC tracks the driver-set velocity. If the relative distance is less than the safe distance, the ACC tracks the minimum of the driver-set velocity and the lead vehicle velocity. The equations for the ACC are as follows:

$$\boldsymbol{V_{ego}} = \begin{cases} \boldsymbol{V_{set}} & \boldsymbol{D_{rel} \geq D_{safe}} \\ \mathbf{min}\,(\boldsymbol{V_{set}}, \boldsymbol{V_{lead}}) & \boldsymbol{D_{rel} < D_{safe}} \end{cases}$$

The safe distance can be found using the following equation:

$$\boldsymbol{D_{safe} = D_{default} + T_{gap} \times V_{ego}}$$

where $\boldsymbol{D_{default}}$is the minimum safe distance and it prevents $\boldsymbol{D_{safe}}$from going to zero if $\boldsymbol{V_{ego} = 0}$. $\boldsymbol{T_{gap}}$ is the time needed for the ego vehicle to get to the lead vehicle when the ego vehicle is moving at a velocity $\boldsymbol{V_{ego}}$.
In the vehicle model, both the lead vehicle and ego vehicle have the following transfer function, G(s), between the input acceleration and output velocity:

$$\boldsymbol{G = \frac{1}{s(0.5s + 1)}}$$

where s is the frequency. G(S) approximates the dynamics of the throttle body and vehicle inertia. We assume that both vehicles are moving in a straight line. The ACC algorithm is summarized in Figure 2.

Our approach uses a pre-trained RL Deep Deterministic Policy Gradient (DDPG) agent that is trained to follow the above ACC logic. Figure 3 and Figure 4 show the ego and lead velocities, safe and relative distances. When the relative distance was greater than the safe distance, the ego velocity tracked the driver-set velocity until the relative distance dropped below the safe distance around $\mathbf{t = 29s}$, at which the ego velocity started tracking the lead vehicle velocity until the lead vehicle velocity was higher than the driver-set velocity at $\mathbf{t = 45s}$, then the ego velocity tracked the driver-set velocity. Then the lead vehicle velocity dropped below the driver-set velocity at $\mathbf{t = 52s}$ and the ego velocity tracked the lead vehicle velocity.

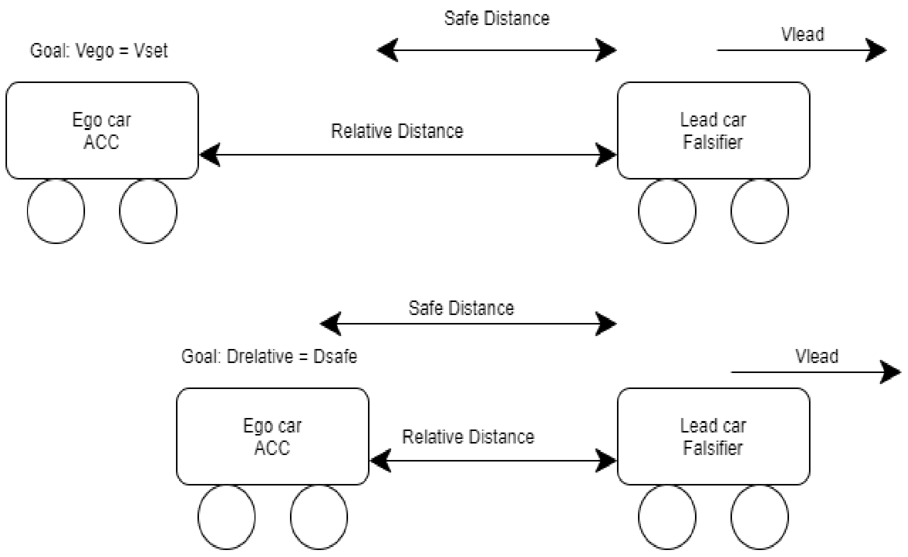

Figure 2. The Adaptive Cruise Control algorithm.

Referring to Figure 1, the ACC would be the agent block and the environment includes the ego vehicle on which it resides and the lead vehicle.

Maintaining a safe distance to the lead vehicle and avoiding a crash are requirements of the ACC. Therefore, the falsifier must test whether the ACC conforms to these requirements. The challenge is to learn a set of initial conditions, namely a relative distance and a relative velocity, and a falsifier acceleration policy that will cause the ACC to violate the safe distance requirement.

The difference between the initial conditions and the acceleration is that initial conditions are calculated once and applied to the system at the beginning of the episode. On the other hand, acceleration is not necessarily fixed during the episode, and it must be calculated on every time step. Thus, it is not practical to use a single RL agent to find initial conditions and acceleration because of the different sampling times at which these variables must be calculated. Therefore, we treat the calculation of the initial conditions and the acceleration separately. We use an agent $\boldsymbol{g_a}$ to calculate acceleration at every time step, and a separate agent $\boldsymbol{g_{IC}}$ to calculate the initial conditions only at the beginning of each episode.

We treat the problem of finding initial conditions as a multi-armed bandit problem and train an agent to choose an initial position and an initial velocity for the lead vehicle from a discretized action space:

$$\begin{gathered}\boldsymbol{Initial\,positions = [100\,,200]\,m} \\ \boldsymbol{Initial\,velocities = [20\,,45]\,m/s}\end{gathered}$$

We discretize the state space in increments of $\boldsymbol{20m}$ for the initial positions and $\boldsymbol{5m/s}$ for the initial velocities. We assume that the accelerations of the ego and lead vehicles $\boldsymbol{a_{ego}}$ and $\boldsymbol{a_{lead}}$, respectively, both lie within $[\boldsymbol{-3,2}]\boldsymbol{m/s^2}$. The ego vehicle starts in the same position and velocity at the beginning of each episode and both vehicles move only in a straight line.

The state vector $\boldsymbol{S_t}$ is composed of the following:

$$\begin{gathered}\boldsymbol{D_{rel} = X_{lead} - X_{ego}} \\ \boldsymbol{V_{rel} = V_{lead} - V_{ego}} \\ \boldsymbol{a_{lead}} \\ \boldsymbol{a_{ego}}\end{gathered}$$

The requirements of the ACC are to maintain a safe distance to the lead vehicle and avoid a crash. Encoding these in a reward function gives the following:

$$\boldsymbol{R_t = 10C + 0.001M - V}$$

where $\boldsymbol{C = 1}$ if a collision has occurred, $\boldsymbol{M = 1}$ if $\boldsymbol{D_{safe} < D_{rel}}$, $\boldsymbol{V = 1}$ if the lead drives backward. While this is not a requirement of the ACC, it is necessary to prevent the falsifier from driving back into the ego vehicle.

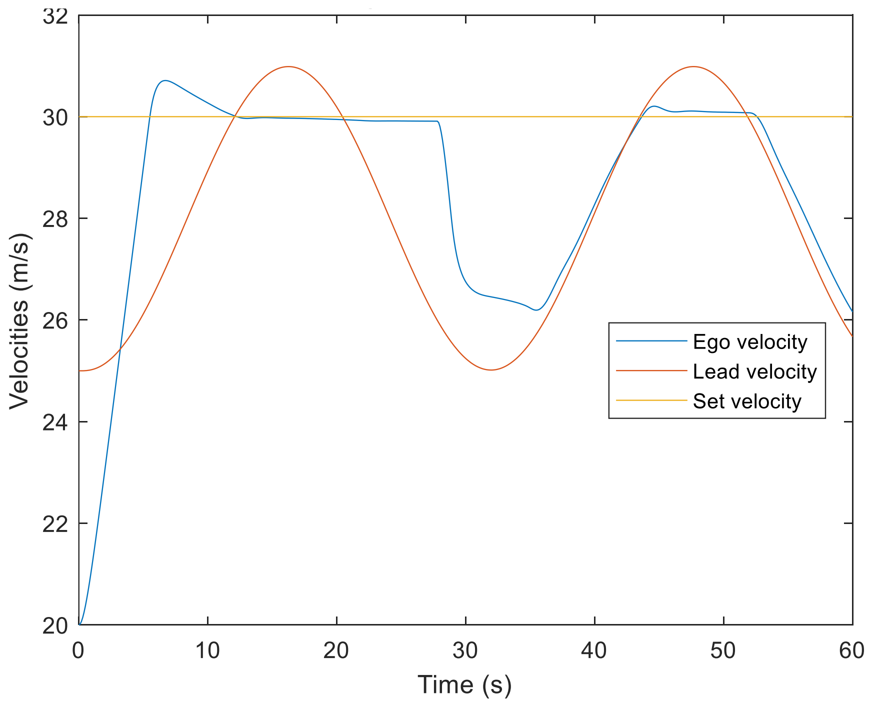

Figure 3. Simulation of the ACC without the falsifier.

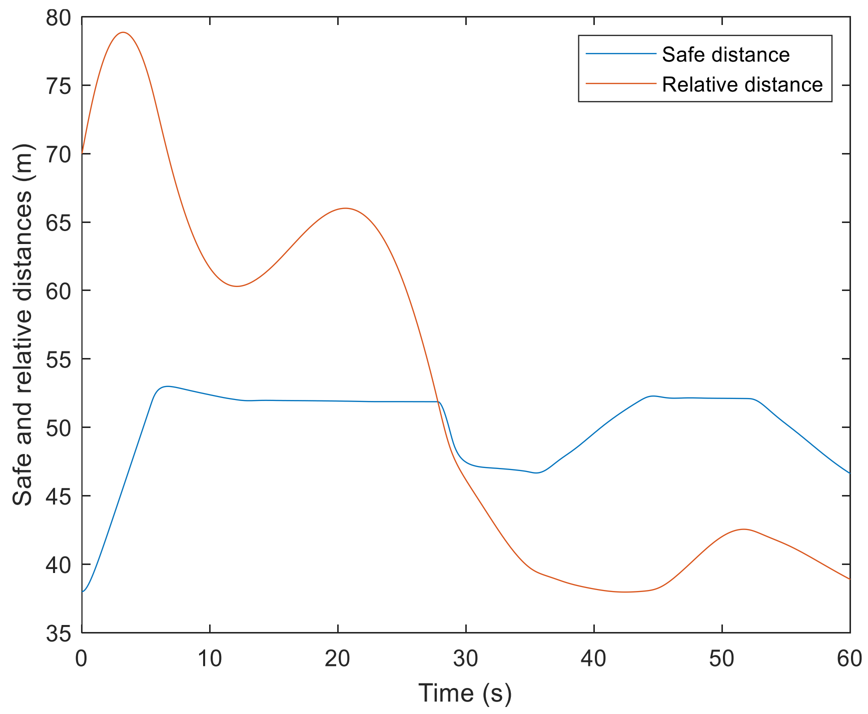

Figure 4. Simulation of the ACC without the falsifier.

Training Algorithm

We use an actor-critic algorithm for agent $\boldsymbol{g_a}$ since it can be trained in an environment with continuous state and action spaces and because of the way the algorithm chooses the actions. The actor takes state $\boldsymbol{S_t}$ and returns a mean and a standard deviation of a normal distribution which will be sampled for an acceleration $\boldsymbol{a_t}$. This promotes agent exploration. The critic takes state $\boldsymbol{S_t}$ and returns the corresponding expectation of the discounted long-term reward $\boldsymbol{V_t}$. We use ε-greedy action selection for agent $\boldsymbol{g_{IC}}$. At the end of each training episode, agent $\boldsymbol{g_{IC}}$ updates its estimated returns for the selected initial conditions based on the received cumulative reward $\boldsymbol{R_c}$ rather than instantaneous reward $\boldsymbol{R_t}$.

The algorithms for g_a and g_{IC} work together as follows:

1. Initialize actor $\mu(S)$ and critic $V(S)$ networks for agent g_a with random parameter values θ_μ and θ_V, respectively and initialize estimated returns $Q(IC)\ \forall\ IC$ for agent g_{IC}.
2. At the start of each episode, choose initial conditions IC

$$IC \leftarrow \begin{cases} argmax_{IC} Q(IC) & with\ probability\ 1-\epsilon \\ a\ random\ action & with\ probability\ \epsilon \end{cases}$$

3. Interact with the environment for N steps using the current policy $\mu(S)$
4. For each step t_s in the N steps:
   a) Compute the return

$$G_t = \sum_{k=t}^{ts+N} (\gamma^{k-t} R_k) + \ b\gamma^{N-t+1} V(S_{ts+N}|\theta_v)$$

   where b is 0 if S_{ts+N} is a terminal state and 1 otherwise and $t =\ t_s + 1, t_s + 2, \dots, t_s + N$ and $t_s = 1$ at the beginning of the training episode.

   b) Compute the advantage function

$$D_t = G_t - V(S_t|\theta_V)$$

c) Compute the gradients for actor network using

$$d\theta_\mu = \sum_{t=1}^{N} \nabla_{\theta_\mu} \ln\,(\mu(S_t|\theta_u) * D_t$$

d) Compute the gradients for critic network using

$$d\theta_V = \sum_{t=1}^{N} \nabla_{\theta_V}\,(G_t - V(S_t|\theta_V))^2$$

e) Update actor and critic parameters using

$$\theta_\mu = \theta_\mu + \alpha d\theta_\mu$$
$$\theta_V = \theta_V + \beta d\theta_V$$

where α and β are the learning rates for the actor and critic, respectively.

5. Calculate the cumulative reward R_c at the end of the episode and update estimated returns for agent g_{IC}: $Q(IC) \leftarrow Q(IC) + \alpha_1[R_c - Q(IC)]$
6. Repeat steps 2 through 5 for each training episode until training is complete.

We use neural networks with two hidden layers and 48 units each with a $\boldsymbol{tanh}$ activation function for agent $\boldsymbol{g_a}$. We use Optimistic Initial Estimates for agent $\boldsymbol{g_{IC}}$ to promote exploration of all the action space.

Table 1 Parameters of the Experiment.

Parameters of the Experiment		
Parameter Name	***Parameter***	***Value***
Minimum safe distance	$\boldsymbol{D_{default}}$	10 m
Time to catch lead vehicle	$\boldsymbol{T_{gap}}$	1.4 s
Ego initial position	$\boldsymbol{P_{Ein}}$	10 m
Ego initial velocity	$\boldsymbol{V_{Ein}}$	20 m/s
Driver set velocity	$\boldsymbol{V_{set}}$	30 m/s
Sampling time	$\boldsymbol{T_S}$	0.1 s
Max episode length	$\boldsymbol{L_{max}}$	60 s
Learning rate of the actor	$\boldsymbol{\alpha}$	8×10^{-4}
Learning rate of the critic	$\boldsymbol{\beta}$	8×10^{-3}
Discount factor	$\boldsymbol{\gamma}$	0.99
Number of steps	$\boldsymbol{N}$	32
Optimistic initial	$\boldsymbol{Q(IC)}$	12

Figure 5 shows the interaction of the falsifier with the environment. The environment includes the ACC and the ego vehicle on which it resides and the lead vehicle. The falsifier agent is both $\boldsymbol{g_a}$ and $\boldsymbol{g_{IC}}$ which reside on the lead vehicle. In other words, the falsifier agent treats the ACC agent as a part of the environment.

PERFORMANCE EVALUATION

For the performance evaluation, MATLAB/Simulink R2021a is used to build the model, train the agents and evaluate the framework (Mathworks, 2024) (Medrano-Berumen et al., 2020). The Reinforcement Learning Toolbox and Deep Learning Toolbox are used to build the actor-critic agent. The training was run for 10,000 episodes, each with 600 maximum time steps, which amounts to a maximum of 60 seconds for each episode. Training was done on a Core i5 processor @2.5GHz with 32GB of RAM.

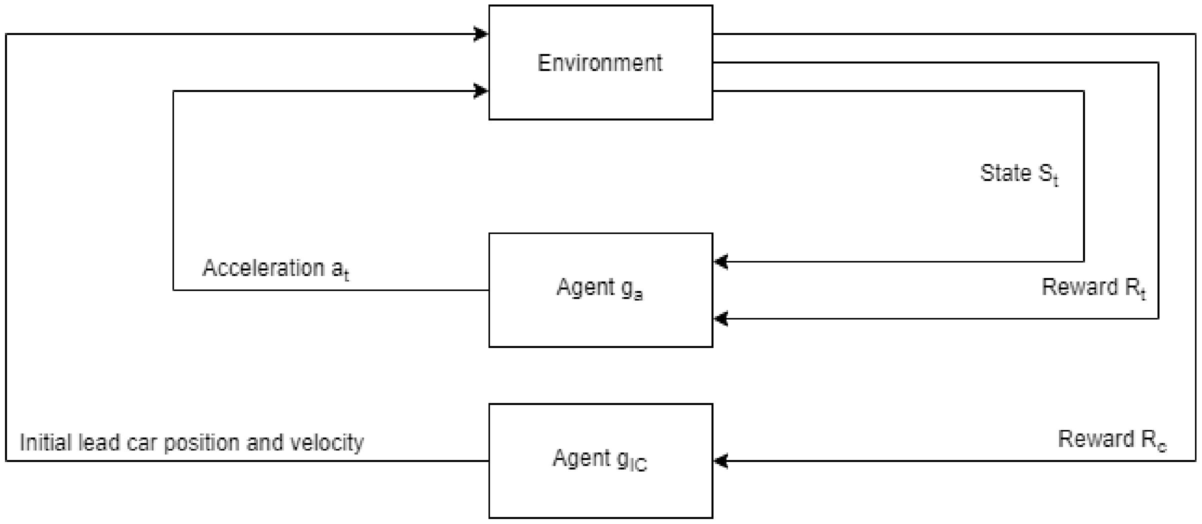

Figure 5 The falsifier interaction with the environment which includes the ACC and the original environment.

Figure *6* shows the cumulative reward averaged over the last 100 episodes. The average cumulative reward increases to about 10 after about 600 training episodes, meaning the agents have learned to falsify the ACC. The noise in the reward is due to low rewards from episodes in which exploratory actions were taken by the agents. Figure *7* shows the cumulative reward for each initial condition averaged over 10,000 episodes.

The lead initial positions between [100-200]m do not significantly affect the returns. However, for initial lead velocities between 20-30 m/s, the return is about 10, meaning the ACC was most likely to fail to avoid collision with the lead vehicle when it starts at those initial velocities. The returns drop to below 10 for initial velocities between 35-40 m/s, meaning the ACC failed sometimes and succeeded at others. For initial lead velocities at 45m/s, the graph suggests that the ACC was least likely to fail, however, that does not necessarily mean that the ACC always succeeds when the lead vehicle starts at those initial conditions as we will see in the simulations.

We simulate the system at three sets of initial conditions, when the initial lead velocity is lower than, equal to, and greater than the set velocity of the ACC, each with a specific initial lead position. The set velocity is selected to be 10m/s=108km/h.

Figure *8* and Figure *9* shows the results of simulating the system with initial lead position and velocity = 100m, 20m/s, respectively. ACC performs well by tracking the set velocity when the relative distance was greater than the safe distance until t = 6s when the relative distance drops below the safe distance and the ACC tracks the lead velocity. Then at t = 11s, the falsifier accelerates to trick the ACC into accelerating before the former brakes again at t = 13.8s, causing the ACC to crash the lead vehicle at t = 15s.

Figure *10* and Figure *11* shows the results of simulating the system with initial lead position and velocity = 150m, 30m/s respectively. We see the same trend from the previous case in which the falsifier accelerates and brakes until the ACC crashes the lead vehicle at t = 42s.

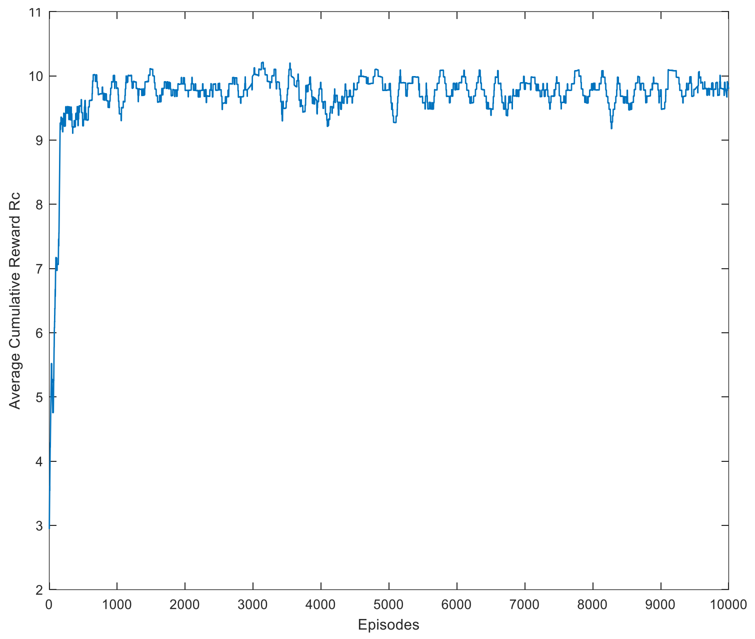

Figure 6. Cumulative reward averaged over the past 100 episodes.

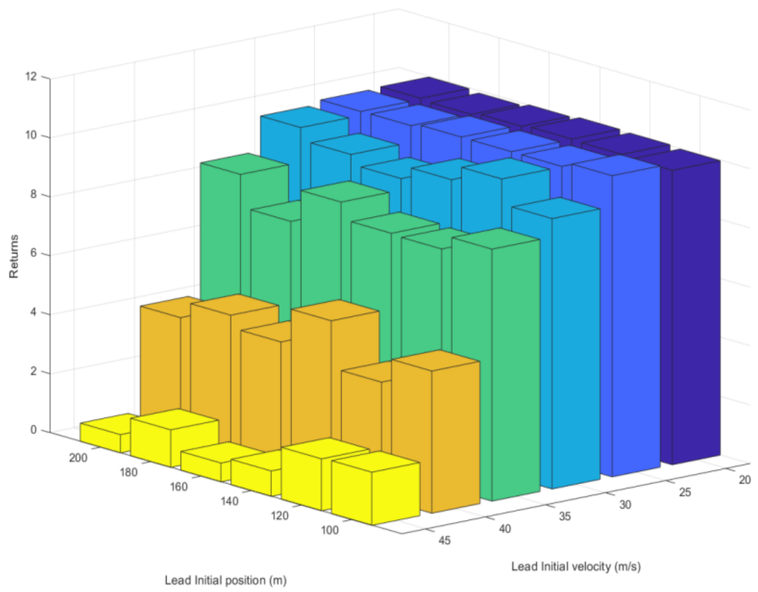

Figure 7. Returns vs. initial conditions for agent g_{IC}.

The low returns in Figure *7* suggest that the falsifier could not succeed in falsifying the ACC at those conditions. However, Figure *12* and Figure *13* show the results of simulating the system for longer than 60 seconds with initial lead position and velocity = 200m, 45m/s, respectively. We see at t = 74s the ACC crashes the lead vehicle. Therefore, we conclude that the falsifier can falsify the ACC when it has enough time to do so.

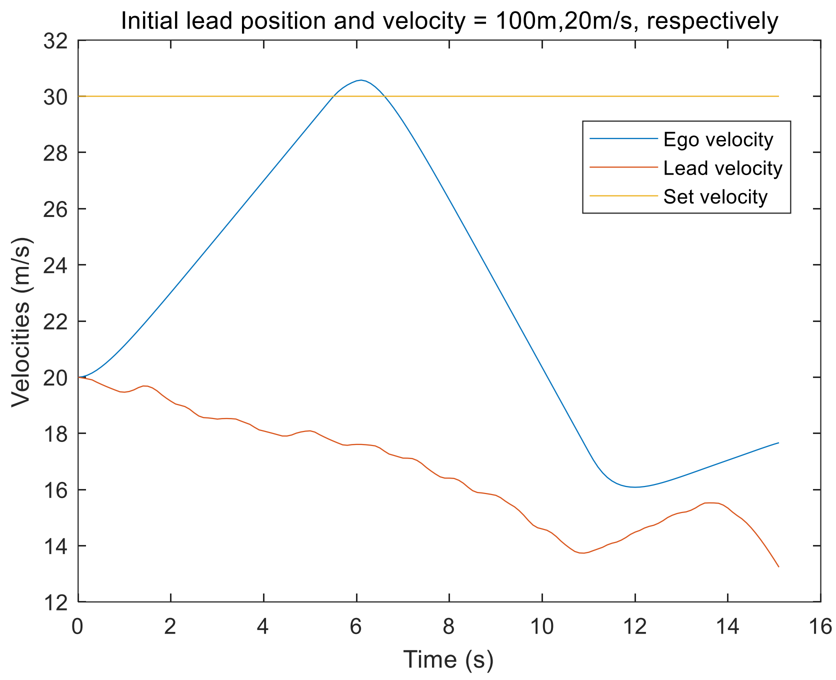

Figure 8. Velocities of Ego and Lead Vehicles vs. Time.

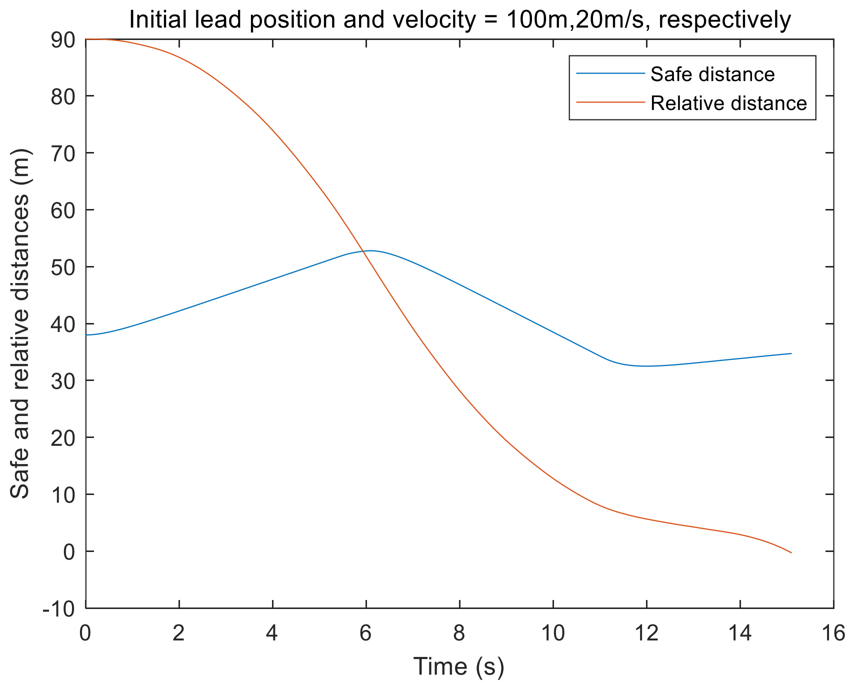

Figure 9. Safe and Relative Distances vs. Time.

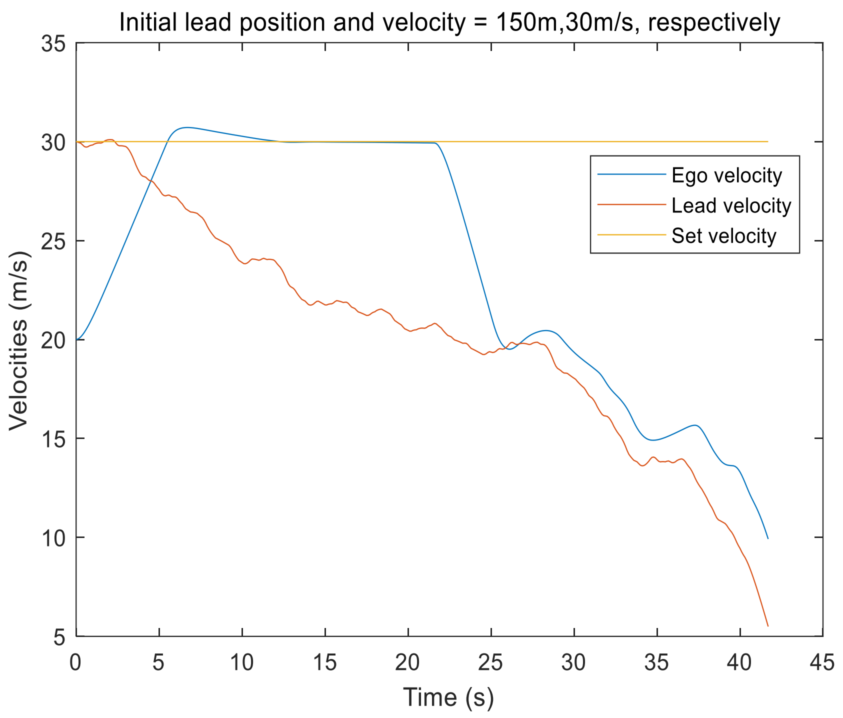

Figure 10. Velocities of Ego and Lead Vehicles vs. Time

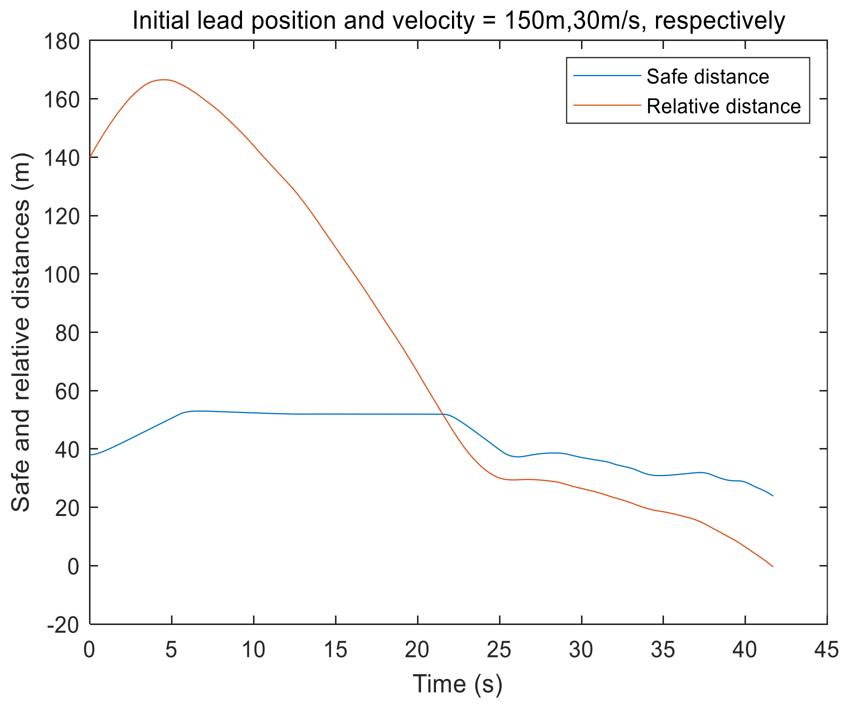

Figure 11. Safe and Relative Distances vs. Time

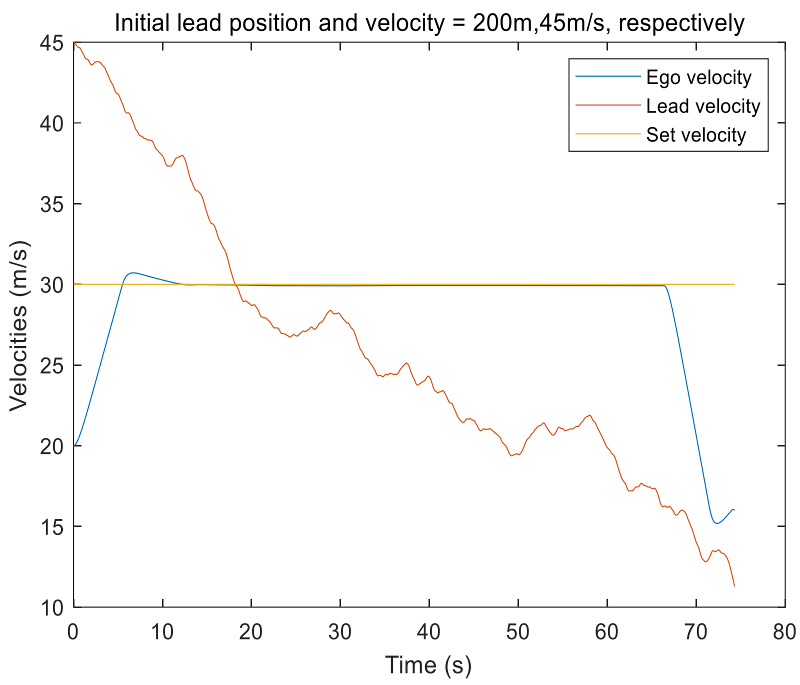

Figure 12. Velocities of Ego and Lead Vehicles vs. Time

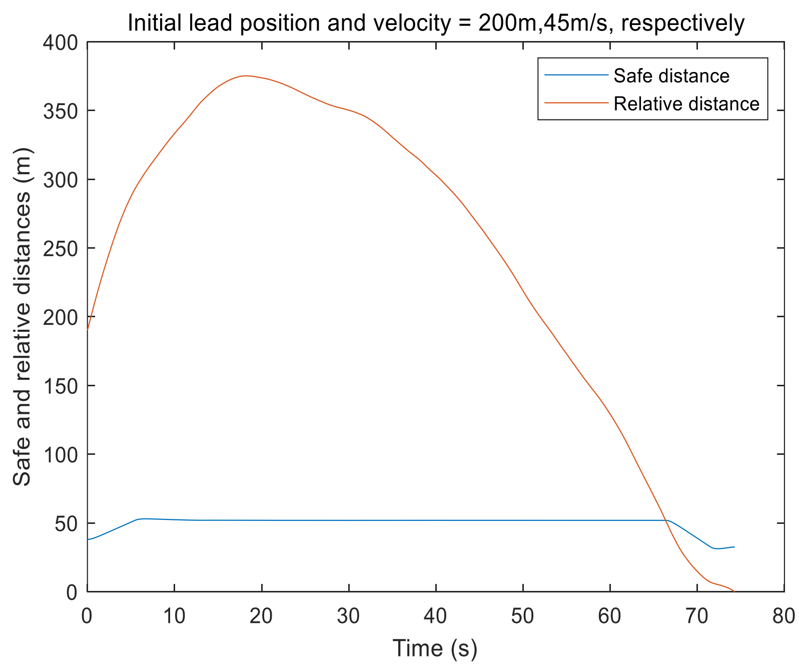

Figure 13. Safe and Relative Distances vs. Time

The results of the performance evaluation show that the proposed approach can be used to improve the design of the ACC. For example, a better ACC performance can be achieved by tuning the safe distance equation parameters. To revalidate the design, the system designer can either simulate the new system with the previously trained falsifier, or continue training the same falsifier, or train another falsifier from scratch. The latter two options would be preferable since tuning the ACC might significantly change the mapping between the states and the actions of the falsifier.

CONCLUSION

In this work, we proposed a framework to use RL in testing cyber-physical systems. Our framework is based on encoding system requirements in the reward function of the falsifier agent. We apply our framework on testing an ACC system and show that it succeeded in identifying a set of inputs to the system that cause it to fail to conform to its safety requirements. This information can be used by the system designers to help design and create a more robust system. Further work can be done by using different safe distance measures (Althoff and Losch, 2016), using a different ACC controller, or by evaluating the performance of different RL algorithms on the task. In addition, it can be extended by applying the same framework to a different problem domain such as Uncrewed Aerial Systems (UAS).

REFERENCES

Akinci, T. C. , Topsakal, O. and Akbas, M. I. Machine Learning Methods from Shallow Learning to Deep Learning. In: Shallow Learning vs. Deep Learning. The Springer Series in Applied Machine Learning. Springer, Cham. https://doi.org/10.1007/978-3-031-69499-8_1, 2024.

Liang, S., Yang, Z., Jin, F., Chen, Y. (2020) Data Centers Job Scheduling with Deep Reinforcement Learning. In: Lauw H., Wong RW., Ntoulas A., Lim EP., Ng SK., Pan S. (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2020. Lecture Notes in Computer Science, vol 12085. Springer, Cham. https://doi.org/10.1007/978-3-030-47436-2_68.

Abdellatif, A. A., Mhaisen, N., Mohamed, A., Erbad, A., and Guizani, M.. "Reinforcement learning for intelligent healthcare systems: A review of challenges, applications, and open research issues." IEEE Internet of Things Journal 10, no. 24 (2023): 21982-22007.falsica.

Fard, Neshat Elhami, Rastko R. Selmic, and Khashayar Khorasani. "A review of techniques and policies on cybersecurity using artificial intelligence and reinforcement learning algorithms." IEEE Technology and Society Magazine 42, no. 3 (2023): 57-68.

Optimizing Chemical Reactions with Deep Reinforcement Learning, Zhenpeng Zhou, Xiaocheng Li, and Richard N. Zare, ACS Central Science 2017 3 (12), 1337-1344, DOI: 10.1021/acscentsci.7b00492.

Deep Reinforcement Learning framework for Autonomous Driving. Sallab, Ahmad EL; Abdou, Mohammed; Perot, Etienne; Yogamani, Senthil. : Society for Imaging Science and Technology, DOI: https://doi.org/10.2352/ISSN.2470-1173.2017.19.AVM-023.

Summer, A., Thompson, J. M. and M. I. Akbas. "CARLoS: Customizable Autonomy Research Low-fidelity Simulator for Safety Testing of Agents." In the IEEE World Forum on Public Safety Technology (IEEE WF-PST), Orlando, Florida, September, 2025.

Andrew Barto, Richard S. Sutton. Reinforcement Learning: An Introduction. A Bradford Book; second edition (March 1, 1998)

Medrano-Berumen, C., & Akbaş, M. İlhan (2020). Validation of decision-making in artificial intelligence-based autonomous vehicles. *Journal of Information and Telecommunication*, *5*(1), 83–103. https://doi.org/10.1080/24751839.2020.1824154

Razdan, R., Akbaş, M. İ., Sell, R., Bellone, M., Menase, M., & Malayjerdi, M. (2023). Polyverif: An open-source environment for autonomous vehicle validation and verification research acceleration. IEEE Access, 11, 28343-28354.

Akbas, M. I. (2021). Testing and validation framework for autonomous aerial vehicles. Journal of Aviation/Aerospace Education & Research, 30(1), 1-19.

Yamagata, Y., Liu, S., Akazaki, T., Duan, Y. and Hao, J., 2020. Falsification of cyber-physical systems using deep reinforcement learning. IEEE Transactions on Software Engineering, 47(12), pp.2823-2840.

Doreste, Andréa, Matteo Biagiola, and Paolo Tonella. "Adversarial testing with reinforcement learning: A case study on autonomous driving." In 2024 IEEE Conference on Software Testing, Verification and Validation (ICST), pp. 293-304. IEEE, 2024.

Cai, X., Bai, X., Cui, Z., Hang, P., Yu, H., & Ren, Y. (2024). Adversarial Stress Test for Autonomous Vehicle Via Series Reinforcement Learning Tasks With Reward Shaping. IEEE Transactions on Intelligent Vehicles.

Lu, Chengjie, Yize Shi, Huihui Zhang, Man Zhang, Tiexin Wang, Tao Yue, and Shaukat Ali. "Learning configurations of operating environment of autonomous vehicles to maximize their collisions." IEEE Transactions on Software Engineering 49, no. 1 (2022): 384-402.

Feng, S., Sun, H., Yan, X., Zhu, H., Zou, Z., Shen, S., & Liu, H. X. (2023). Dense reinforcement learning for safety validation of autonomous vehicles. Nature, 615(7953), 620-627.

Vahid, B. and Munir, A. "Adversarial reinforcement learning framework for benchmarking collision avoidance mechanisms in autonomous vehicles." IEEE Intelligent Transportation Systems Magazine 13, no. 2 (2019): 236-241.

X. Qin, N. Aréchiga, J. Deshmukh and A. Best, "Robust Testing for Cyber-Physical Systems using Reinforcement Learning," 2023 21st ACM-IEEE International Symposium on Formal Methods and Models for System Design (MEMOCODE), Hamburg, Germany, 2023, pp. 36-46.

Koren, Mark, Saud Alsaif, Ritchie Lee, and Mykel J. Kochenderfer. "Adaptive stress testing for autonomous vehicles." In 2018 IEEE Intelligent Vehicles Symposium (IV), pp. 1-7. IEEE, 2018.

M. Treiber, A. Hennecke, and D. Helbing, “Congested traffic states in empirical observations and microscopic simulations,” Physics Review E, vol. 62, pp. 1805–1824, Aug 2000.

Wang, Xiao, Saasha Nair, and Matthias Althoff. "Falsification-based robust adversarial reinforcement learning." In 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 205-212. IEEE, 2020.

The MathWorks, Inc. “Statistics and Machine Learning Toolbox Documentation” mathworks.com. Accessed: December 07, 2024. Available: https://www.mathworks.com/help/stats/index.html.

Medrano-Berumen, C., & Akbaş, M. İ. (2020, March). Scenario generation for validating artificial intelligence based autonomous vehicles. In Asian Conference on Intelligent Information and Database Systems (pp. 481-492). Cham: Springer International Publishing.

M. Althoff and R. Losch. Can automated road vehicles harmonize with traffic flow while guaranteeing a safe ¨ distance? In Proc. of the 19th International IEEE Conference on Intelligent Transportation Systems, pages 485–491, 2016.