Hindawi
Journal of Advanced Transportation
Volume 2023, Article ID 3187944, 10 pages
https://doi.org/10.1155/2023/3187944

Research Article

Using Reinforcement Learning to Handle the Unintended Lateral Attack in the Intelligent Connected Vehicle Environment

Luoyi Huang,[1,2] **Wanjing Ma**,[1] **Ling Wang**,[1] **and Kun An**[1]

[1]*The Key Laboratory of Road and Traffic Engineering, Ministry of Education, Tongji University, Shanghai 201804, China*
[2]*Bosch Automotive Products (Suzhou) Co. Ltd., Suzhou 215025, China*

Correspondence should be addressed to Wanjing Ma; mawanjing@tongji.edu.cn

Received 16 August 2022; Revised 15 October 2022; Accepted 17 March 2023; Published 21 April 2023

Academic Editor: Wenxiang Li

It is widely accepted that an unintended lateral attack is inevitable in the intelligent connected vehicle environment. This paper explores the feasibility of a reinforcement learning method named PPO (Proximal Policy Optimization) to handle the unintended lateral attack and keep the vehicle stay in the ego lane. Based on the China highway design guide, the discrete speed variants of 120 km/h, 100 km/h, and 80 km/h were selected, along with different curvatures ranging from 250 m to 1200 m in every 50 m as combinations of speed-curvature test. The tests were implemented in the Open.ai CarRacing-v0 simulation environment with an external racing wheel attached to simulate the unintended lateral attack. The simulation results show that the PPO can handle the unintended lateral attack on the standard-designed highway in China. The results can be applied to the intelligent connected vehicle to be mass-produced in the future.

1. Introduction

Intelligent connected vehicle is shaping the automotive industry. It allows the vehicle to communicate with other traffic participants. In a connected environment, the attack is inevitable. It has many possible ways to handle the identified attack from a security perspective [1]. However, it has been found to be only a few studies on the unintended attack from a functional safety perspective.

Along with the rapid development of the intelligent connected vehicle, safety is becoming an important issue to consider. Automotive Functional Safety (ISO 26262, Road vehicles-Functional safety [2]) has become the de facto practice for intelligent connected vehicle to be produced in the market. ISO 26262 generally gives a system credit for a human driver ultimately being responsible for safety, which consists of three evaluation factors: severity, the probability of exposure, and controllability [3]. ISO 26262 is explicitly targeted for automotive safety, providing a safety lifecycle that includes development, production, operation, service, and decommissioning. ISO 26262 defines the ASIL (Automotive Safety Integrity Level). ASIL is calculated from severity, probability of exposure, and controllability. As of today, there is no fully autonomous vehicle that end-users can buy in the market, and one reason is that absolute safety cannot be proved in a commonly accepted way.

Functional safety cares about the Electric/Electronic malfunction behavior of the vehicle. Its practice has evolved for many years and has already been applied in mass-produced vehicles. SOTIF, which stands for Safety of the Intended Functionality, is a logical supplement to the established functional safety standard ISO 26262. SOTIF deals with the functional limitation of the vehicle concerning the absence of unreasonable risk due to hazards resulting from functional insufficiency of the intended functionality together with the reasonably foreseeable misuse by persons [4]. SOTIF is currently difficult to quantify. Take the example of lines of source code. Air force F-22 has around 1.7 million lines of source code, while Boeing 787 has 6.5 million lines and air force F-36 has 24 million lines. Compared with these examples, the luxury vehicle already mass-produced in the market has 100 million lines [5]. Based on the experts' assumption, the lines of source code of autonomous vehicle will increase exponentially considering the complexity of the

functionality. With this tremendous amount of source code increased, there is a higher risk of an autonomous vehicle to be failed in some corner cases. In addition, with the introduction of deep learning technology, it can be more challenging to bring unknown compared to the traditional Vee model development. To address this, García and Fernández [6] introduced the Safe Reinforcement Learning, which was defined as the process of learning policies that maximize the expectation of the return in problems in which it is crucial to ensure reasonable system performance and/or respect safety constraints during the learning and/or deployment processes.

Besides safety, security is also an essential factor to consider in the intelligent connected vehicle since intelligent connected vehicle connect with other vehicles, infrastructure, and the cloud. A vehicle is no longer an isolated object in an intelligent connected vehicle environment. Along with the enriched functionality enabled by connectivity, the vehicle opens the attack interface for external resources. Dibaei et al. [7] summarized common attack methods in the intelligent connected vehicle environment, mainly including DoS (Denial of Service), DDoS (Distributed Denial-of-service), black-hole attack, replay attack, Sybil attack, impersonation attack, malware, falsified information attack, and timing attack. Even with modern encryption technology, security vulnerabilities can still be found in the automotive industry. Chattopadhyay et al. [8] found that the security-by-design principle for autonomous vehicle is poorly understood and rarely practiced. The intelligent connected vehicle is prone to attack, and unintended attack is inevitable. The assumption was made in this paper that attack exists and cannot be eliminated. In addition, the feet-free longitudinal function has been widely studied and released in the market for years like adaptive cruise control, so the unintended lateral attack was focused on in this paper.

The unintended lateral attack is not only an automotive issue, but it can also cause an environmental problem and become a barrier to achieving low-carbon transportation [9]. Therefore, research on "how to handle the unintended lateral attack" is a must to reach the future intelligent transportation systems. One possible way to handle unintended lateral attack is reinforcement learning. Reinforcement learning is being used by an agent to learn behavior through trial-and-error interactions with the environment [10]. A standard reinforcement learning model is shown in Figure 1.

An agent is connected to the environment through perception and action. At each step of the interaction, the agent receives an input: i, with the indication of the current state: s, selects an action: a, generates an output. The action changes the state of the environment, and the value of this state transition is communicated to the agent via a scalar: r. The agent's behavior B, is targeting to select actions that can increase the long-run sum of rewards. The agent can learn to do this over time through trial-and-error interactions. In recent years, reinforcement learning has been applied in the game of Go [11], highly automated driving [12–14], traffic signal control [15–17], and has proven its effectiveness. Meanwhile, multi-agent reinforcement learning is becoming

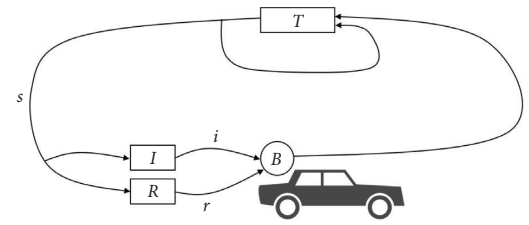


Figure 1: Reinforcement learning model. The vehicle interacts with the environment via trial-and-error method.

a hot research area [18–20]. Whatever single-agent or multiagent reinforcement learning, there are some basic and commonly used reinforcement learning methods like DQN (Deep Q Networks), PG (Policy Gradient), DDPG (Deep Deterministic Policy Gradient), TD3 (Twin Delayed DDPG), SAC (Soft Actor Critic), and A2C (Advantage Actor Critic). In 2017, OpenAI published a novel objective function that enables multiple epochs of minibatch updates named PPO (Proximal Policy Optimization) [21], a family of policy optimization methods, which achieved a favorable balance between sample complexity, simplicity, and wall-time. The PPO algorithm is illustrated as follows:

for iteration = 1, 2, . . . , **do**
  for actor = 1, 2, . . . , N **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
    Compute advantage estimates $\widehat{A}_1, \ldots, \widehat{A}_T$
  end for
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$
  $\theta_{\text{old}} \leftarrow \theta$
end for

PPO uses two neural networks: the policy $\pi(s)$ and the value function $V(s)$. The policy $\pi(s)$ maps an observation s_t to an action a_t, while the value function $V(s)$ maps an observation s_t to a scalar value showing how advantageous it is to be in that state. The value network estimates the value of each state by minimizing the error between the predicted value and the actual value. The policy network uses the estimate of value function to select actions that lead to higher rewards.

Resulting from these considerations, the remainder of this paper is organized as follows. Section 2 introduces the methods used in this study, including test scenario design, simulation environment construction, training procedure, and attack injection logic. Section 3 presents the positive simulation results and illustrates the effectiveness of our method. Section 4 concludes the findings and identifies open areas of research for future work.

2. Methods

The test scenarios were defined based on the standard of the highway in China. A modified CarRacing-v0 simulation environment was used to generate the test scenarios and provide a secondary development interface for

reinforcement learning implementation. The PPO algorithm was then applied to the selected scenarios for training; afterwards, the trained models were used to infer the rest of the test scenarios. The unintended lateral attack was simulated by attaching an external driving force racing wheel.

2.1. Test Scenario Design. The test scenario is the combinations of speed and curvature on the highway. Based on the "Technical Standard of Highway Engineering" [22] and "Design Specification for Highway Alignment" [23], see Table 1. The most common speed limits on the highway in China are 120 km/h on the standard-designed highway, 100 km/h on the class-1 highway, and 80 km/h on the class-2 highway. A design speed of less than 80 km/h is not a standard highway in China. Therefore, the minimum speed considered in this paper is 80 km/h. For curvature, four specific numbers were identified as follows: 250 m, 400 m, 650 m, and 1200 m. The reasons for selecting these numbers are as follows:

(i) 250 m: the minimum curvature on the standard highway in China. This usually appears at on-ramp and off-ramp

(ii) 400 m: the minimum curvature of highway, which has a speed limit of 100 km/h

(iii) 650 m: the minimum curvature of highway, which has a speed limit of 120 km/h

(iv) 1200 m: the threshold of curvature, which can cover 95% of highway in the Yangtze River area in China

The lane width was set to 3.75 m in our test scenario considering the speed variants were 120 km/h, 100 km/h, and 80 km/h. The test matrix was defined in Table 2.

Take the 100 km/h case for example, curvatures were selected from the range [250, 1200] in every 50 m, thus the following list of curvature can be derived: (250, 300, 350, 400, 450, 500, 550, 600, 650, 700, 750, 800, 850, 900, 950, 1000, 1050, 1100, 1150, 1200).

2.2. Simulation Environment Construction. Simulation is commonly used as an environmental tool to train reinforcement learning algorithms. The results have the potential to be transferred to solve real-world problems [24]. In this paper, CarRacing-v0 was selected as the simulation environment. CarRacing-v0 is a reinforcement learning environment developed by OpenAI [25] to support continuous control. It provides a bird-view racing environment that fits well for the future infrastructure-supported automated driving environment, which differs from the in-vehicle sensing perspective. CarRacing-v0 provides a state, which consists of 96×96 pixels.

The CarRacing-v0 reward is -0.1 every frame and $+1000/N$ for every track tile visited, where N is the total number of tiles in track. According to the example on the official website, if we have finished in 732 frames, the reward is $1000 - 0.1 \times 732 = 926.8$ points. The episode finishes when all tiles are visited.

Table 1: Design of curvature and lane width in "Design Specification for Highway Alignment."

Design speed (km/h)	Limit curvature value (m), with $I = 8\%$, where I donates maximum superelevation value	Lane width (m)
120	650	3.75
100	400	3.75
80	250	3.75
60	125	3.5
40	60	3.5

Due to the limitation of the reward calculation in CarRacing-v0, the vehicle's exact position and the distance between the vehicle and the lane markers are unknown to us. Therefore, in this paper, the following criteria were defined to decide whether reinforcement learning can handle the unintended attack or not.

(i) PASS: the vehicle can move back into the ego lane after the unintended attack; 10 out of 10 succeed

(ii) FAIL: the vehicle cannot move back into the ego lane and leaves the lane ultimately; >1 out of 10 failed

The version of CarRacing-v0 is 0.18.3, which was released in May 2021. The course shape, lane width, and traveling speed were modified accordingly to meet our test requirements. The course in original CarRacing-v0 was randomly generated for reinforcement training and testing. In this paper, the source code was modified and recompiled to generate the fixed shape of the course. Figure 2 shows the randomly generated courses and Figure 3 shows the fixed shape generation after code modification.

In addition to the course shape, the lane width was changed to 3.75 m according to the needs of our study, which is shown in Figure 4.

CarRacing-v0 provides the observation and action control interfaces in a Box manner. Box represents the Cartesian product of n closed intervals, it is the specific type defined by OpenAI gym. The reward range of CarRacing-v0 is (−inf, inf). The actions are in a discrete vector shown in Table 3.

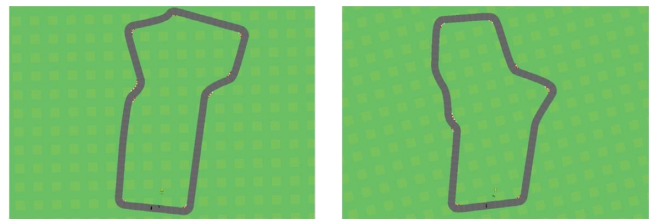
With the different combinations of the action elements, the following action spaces can be derived in Table 4.

To simplify the action and avoid causing the vehicle to drift, only the single action from each action space was selected, marked in bold in Table 4. They are steer left: [−1, 0, 0], no action: [0, 0, 0], accelerate: [0, 0, 1], brake: [0, 0.5, 0], and steer right: [1, 0, 0]. In addition, to better monitor the parameters in real time, the label texts were added in the following order, from left to right: reward, ABS sensor, speed, wheel angle, and angular velocity, which is shown in Figure 5.

2.3. Training Logic and Parameter Setting. The combination space of speed versus curvature is too large; thus, the following combinations were selected for initial training:

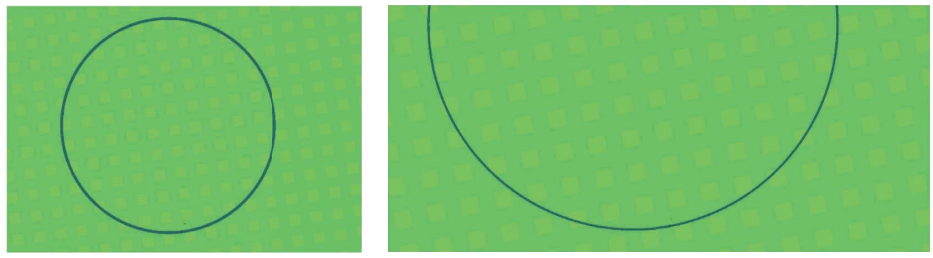
Table 2: Test matrix between speed and curvature.

Speed (km/h)	Curvature (m)
120	Range [250, 1200] : 50, i.e., select curvature from 250 m–1200 m in every 50 m
100	Range [250, 1200] : 50, i.e., select curvature from 250 m–1200 m in every 50 m
80	Range [250, 1200] : 50, i.e., select curvature from 250 m–1200 m in every 50 m



(a) (b)

Figure 2: Randomly generated courses in CarRacing-v0. (a) Random course generation example 1. (b) Random course generation example 2.



(a) (b)

Figure 3: Generation of fixed shapes after code modification. (a) Fixed curvature of 500 m. (b) Fixed curvature of 1700 m.

(a) (b)

Figure 4: Lane width modification. (a) Original lane width. (b) Modified lane width to 3.75 m.

Table 3: Action element.

Action element	Action type	Data range	Meaning of vector
1	Steering	[−1, 0, 1]	[Steer left, no action, steer right]
2	Braking	[0, 0.5]	[No action, brake]
3	Acceleration	[0, 1]	[No action, accelerate]

(120 km/h, 650 m), (100 km/h, 400 m), and (80 km/h, 250 m). The training and inference logic is illustrated in Figure 6.

In the training session shown on the left-hand side in Figure 6, the three separated models for (120 km/h, 650 m), (100 km/h, 400 m), and (80 km/h, 250 m) were trained as the base. According to the definition-of-done from the CarRacing-v0 leaderboard, the "solving" is defined as getting the average reward of 900+ over 100 consecutive episodes, which indicates that the reinforcement learning based in-lane driving has been achieved. After that, the training session finished. In the inference session, the trained model from (120 km/h, 650 m) was used to test the variant of

Table 4: Action space.

Action space	Action combination
1	**[−1, 0, 0]**
2	[−1, 0, 1]
3	[−1, 0.5, 0]
4	[−1, 0.5, 1]
5	**[0, 0, 0]**
6	**[0, 0, 1]**
7	**[0, 0.5, 0]**
8	[0, 0.5, 1]
9	**[1, 0, 0]**
10	[1, 0, 1]
11	[1, 0.5, 0]
12	[1, 0.5, 1]

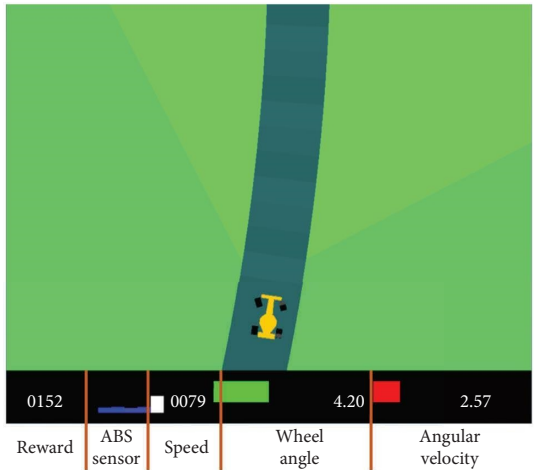


Figure 5: Modified display of parameters in real time. Showing reward/ABS sensor/speed/wheel angle/angular velocity information.

(120 km/h, curvature$_y$), where curvature$_y$ donates the number in the curvature list [250, 1200] : 50. The same logic applies to the speed variants of 100 km/h and 80 km/h.

The architecture of the convolutional neural network is illustrated in Figure 7. The architecture consists of 6 convolutional layers. From the left, the input of RGB image is of 96×96 pixels. The grass in the picture was removed to reduce the complexity since the grass is not crucial in our case. The RGB image was converted to a single gray channel to further reduce the input dimension from three to one.

Every four frames were used to generate actions. Therefore, the input to the neural network was $96 \times 96 \times 4$, where 4 denotes the four continuous frames. Followed by the convolutional layers of $47 \times 47 \times 8$, $23 \times 23 \times 16$, $11 \times 11 \times 32$, $5 \times 5 \times 64$, $3 \times 3 \times 128$, and $1 \times 1 \times 256$ [26], ReLU was used as the activation function. Kingma and Ba [27] was used as the optimizer. Mean squared error loss was used to optimize the difference between the predicted value and the actual value of each state. The clipped loss function was used to limit the probability change that may occur in a single step.

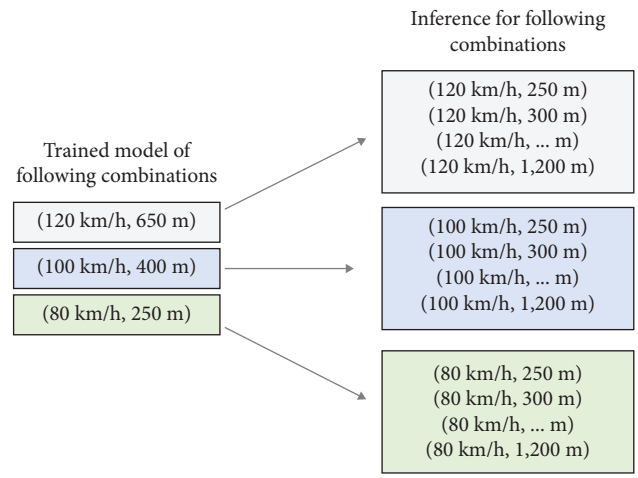


Figure 6: Training and inference logic. Train the reference model as a base and apply the reference model to different scenarios.

The parameters used in the training session:

gamma (discount factor) = 0.99

gae_lambda = 0.95

image stack = 4

max_grad_norm = 0.5

epoch = 10

batch_size = 128

learning rate = $1e-3$

value_coef = 0.5

entropy_coef = 0.01

The training infrastructure was using AMD Ryzen R9-4900HS, 16G DRAM, and Nvidia RTX 2060 Max-Q with cudnn 10.2. PyTorch was used to generate the neural network models.

2.4. Unintended Lateral Attack Injection. To better simulate the unintended lateral attack, a Logitech G29 driving force racing wheel was used as the attack input instead of using pure software button simulation. G29 provides a 900-degree steering angle, in a real-world scenario, especially in highly automated driving, the steering torque or steering angle is limited to a value due to functional safety requirements. The most significant steering value was implemented for unintended lateral attack, that is, −1.0 for the left and +1.0 for the right. The attack injection lasted for 100 milliseconds. The short period implies a sudden action and indicates the most-common calculation cycle from perception to vehicle motion control.

Figure 8 shows the connection between the simulation environment and the attack injection input. In Step 1, the test vehicle was running in the CarRacing-v0 using the trained model in Figure 6. The vehicle can keep itself to drive in the ego lane. In Step 2, a test driver triggered a sudden steering force using the G29, the simulated unintended lateral attack was passed to the vehicle via python SDK in the

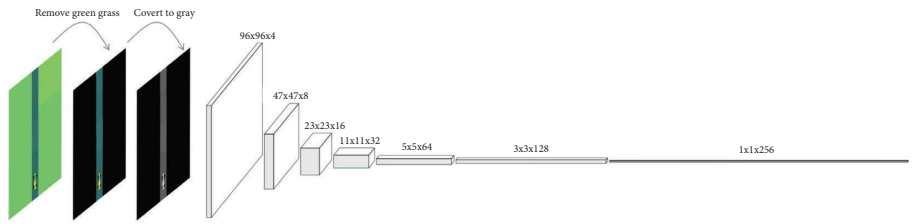


Figure 7: Neural network architecture used in training from image color handling to neural network design.

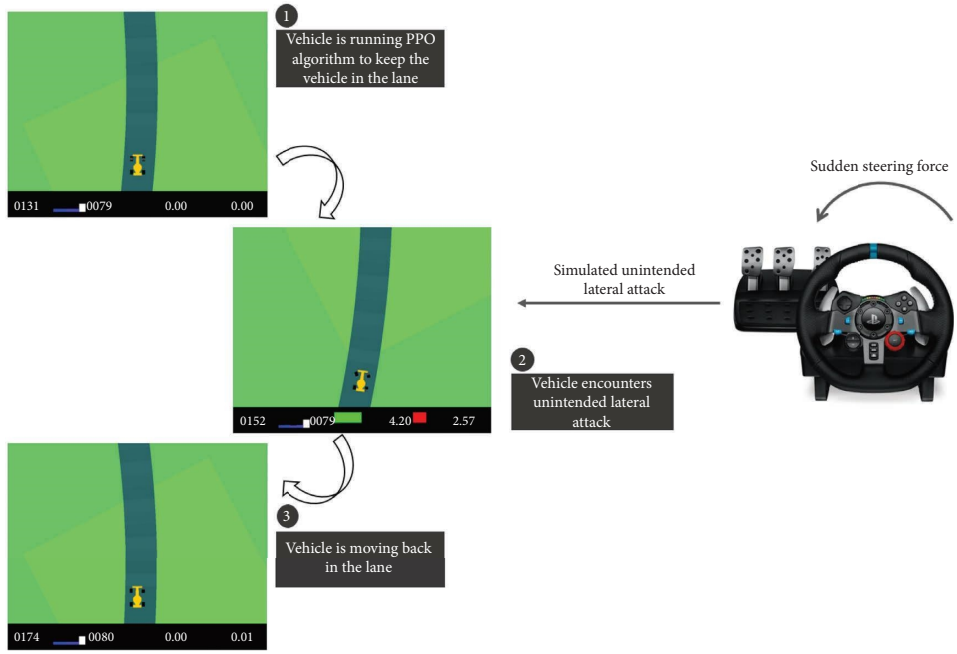


Figure 8: Simulated unintended lateral attack. Use external driving force to simulate the unintended lateral attack.

CarRacing-v0 environment. In Step 3, the vehicle's movement was observed to check whether the PPO algorithm could bring the vehicle back in the lane.

The injection was unintended for the vehicle in the CarRacing-v0 environment, which means the vehicle did not know when the injection would occur. Ten test drivers were invited to trigger the unintended lateral attack injection using G29, data and video were recorded for analysis.

3. Results and Discussion

As outlined in Figure 6, in-lane driving must first be achieved by training and models must be applied to handle unintended lateral attack. The training lasted for around 2 hours for each scenario: (120 km/h, 650 m), (100 km/h, 400 m), and (80 km/h, 250 m). The training episode versus reward and the fitted curve using the logistic fitting method are illustrated in Figure 9. After 200 training episodes, the agent achieved a mean score of 900+ over the next 100 episodes in the three scenarios, reached the "solving" state defined by the CarRacing-v0 leaderboard.

From Figure 9, three turning points were identified from the training results; they are (197, 901) in the curve of (120 km/h, 650 m), (159, 1000) in the curve of (100 km/h, 400 m), and (148, 918) in the curve of (80 km/h, 250 m). After these turning points, the rewards can reach a relatively stable number above 900 and achieve a mean value of 900+ over the next 100 episodes. The mean value of these 100 episodes is calculated in Figure 10.

In the scenario of (120 km/h, 650 m), the mean value reaches 927.19. In the scenario of (100 km/h, 400 m), the mean value reaches 955.32, and in the scenario of (80 km/h, 250 m), the mean value reaches 929.35. All mean values are greater than 900.

The "solving" state was achieved in only 200 training episodes in this paper. The typical number is compared to reach the "solving" state from the CarRacing-v0 leaderboard: 5000, which has 25 times difference. It could be inferred that randomly generated courses increased the complexity of the training. To further explore the situation after 200 training episodes, the result of the (100 km/h, 400 m) scenario in a consecutive 5000 training episodes was drawn in Figure 11.

The curve was expected to become stable after 200 training episodes; however, the curve sharply dropped from episode 340 and started rising again. This recovered training ramp cost 1415 episodes, increased 7 times compared with the first ramp-up in Phase 1. When the reward reached 900+ again in Phase 2, the reward stayed high for around 1100

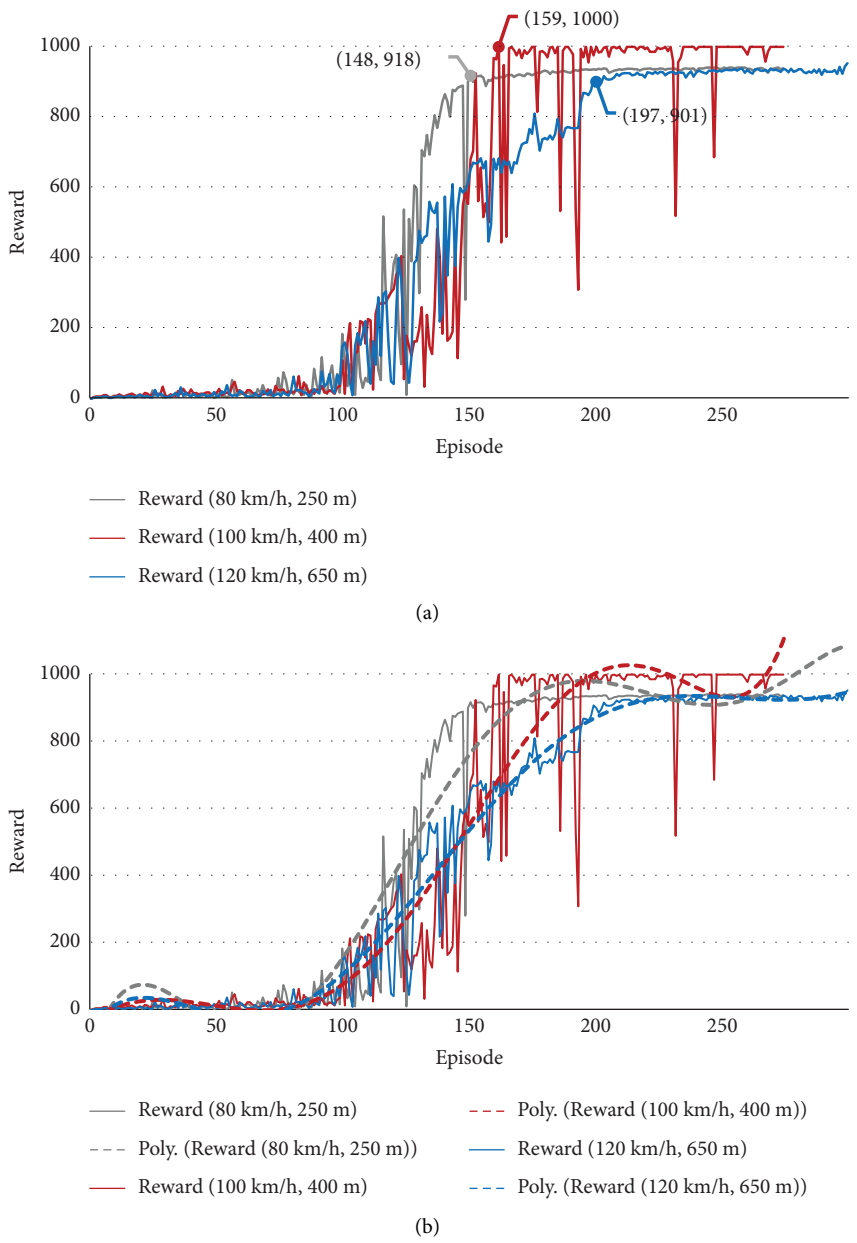


Figure 9: Training result: episode versus reward. (a) Episode versus reward with turning points. (b) Fitted episode versus reward.

episodes and started to drop again at episode 2931. In Phase 3, it cost 502 episodes to reach the "solving" state again at episode 3433 and stayed high for the rest in a relatively stable state. This curve indicates that the reward can still fluctuate with time even when the "solving" state is reached.

Based on the logic described in Figure 6, the trained models were applied to other scenarios for the unintended lateral attack tests. The test results are shown in Figure 12.

120 km/h case was shown at the upper part of Figure 12. The trained model of (120 km/h, 650 m) was applied as reference model in the curvature variants of 250, 300, 350, 400, 450, 500, 550, 600, 700, 750, 800, 850, 900, 950, 1000, 1050, 1100, 1150, and 1200 for unintended lateral attack tests. All combinations passed the tests except (120 km/h, 450 m), (120 km/h, 400 m), (120 km/h, 350 m), (120 km/h, 300 m), and (120 km/h, 250 m). The same logic was applied in scenarios of 100 km/h and 80 km/h. All combinations passed the tests except (100 km/h, 300 m) and (100 km/h, 250 m). The results show that our trained reference model can cover 88% of the unintended lateral attacks listed in this paper.

Taking into account the design standard in China listed in 1, trained models are the worst cases of different speed variants on the highway. If the model can handle the case of (120 km/h, 650 m), the model can also handle the cases of (120 km/h, curvature$_y$) theoretically, where curvature$_y$ is larger than 650 m. According to the design guide in China, the combinations of failed cases (120 km/h, 450 m), (120 km/h, 400 m), (120 km/h, 350 m), (120 km/h, 300 m), (120 km/h, 250 m), (100 km/h, 300 m), and (100 km/h, 250 m) do not

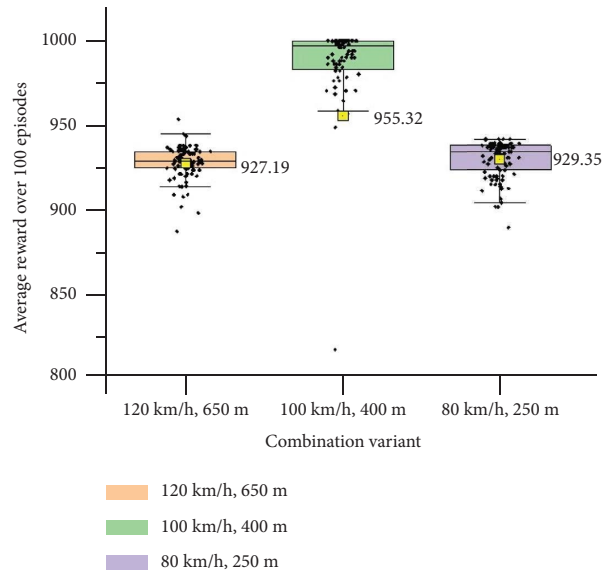


Figure 10: Reward statistics of 3 scenarios. The mean value of the 100 episodes shown before the "solving" state.

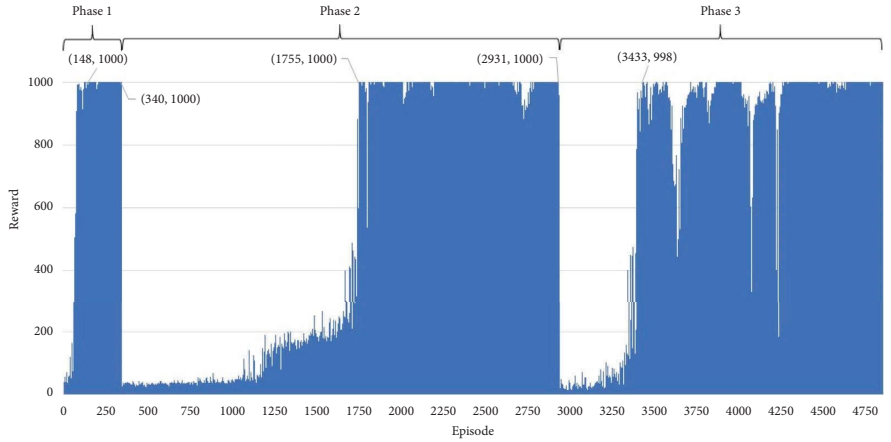


Figure 11: Consecutive 5000 training episodes of the scenario (100 km/h, 400 m). The fluctuation curves are shown in 3 phases.

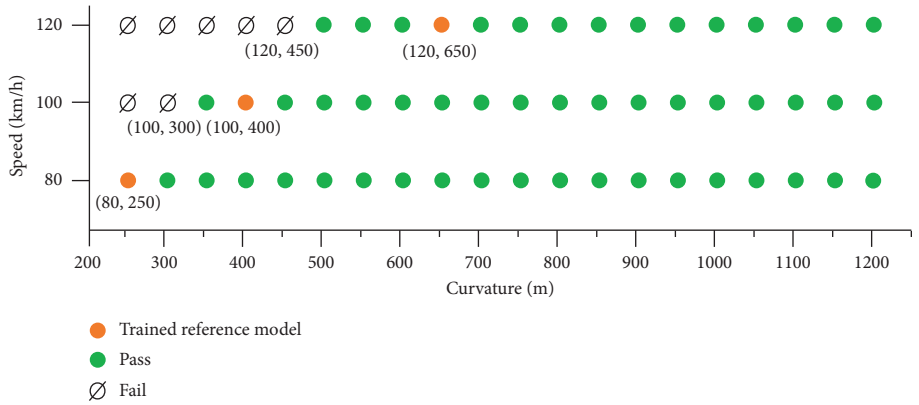


Figure 12: Simulated unintended lateral attack in 3 scenarios: 120 km/h, 100 km/h, and 80 km/h.

exist in the real world. It seems, therefore, that the results can be applied to the standard-designed highways in China.

4. Conclusions

This paper proved the feasibility of PPO reinforcement learning to keep the vehicle in lane driving on the standard-designed highway in China. In addition, PPO can handle the unintended lateral attack and bring the vehicle back in the ego lane in the scenarios of (120 km/h, 500 m to 1200 m), (100 km/h, 350 m to 1200 m), and (80 km/h, 250 m to 1200 m). The results were achieved using the modified CarRacing-v0 simulation environment.

However, this paper trains different models in three different scenarios. It is not the best practice in the real world, which may bring an overfitting problem. In the future, the feasibility of using a single model to cover all scenarios on the real-world highway will be studied.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

This study was supported by the National Natural Science Foundation of China (no. 52131204) and Bosch Automotive Products (Suzhou) Co., Ltd.

References

[1] P. Goyal, S. Batra, and A. Singh, "A literature review of security attack in mobile ad-hoc networks," *International Journal of Computer Application*, vol. 9, no. 11, pp. 11–15, 2010.

[2] Iso, "ISO 26262-1:2018 Road vehicles - functional safety - Part 1: vocabulary," 2018, https://www.iso.org/obp/ui/#iso:std:iso:26262:-1:ed-2:v1:en.

[3] P. Koopman and M. Wagner, "Autonomous vehicle safety: an interdisciplinary challenge," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 90–96, 2017.

[4] M. Khatun, M. Glaß, and R. Jung, "Scenario-based extended HARA incorporating functional safety & SOTIF for autonomous driving," in *Proceedings of the 30th European Safety and Reliability Conference and 15th Probabilistic Safety Assessment and Management Conference*, pp. 53–59, Singapore, January 2020.

[5] D. McCandless, "Codebases: millions of lines of code," 2022, https://www.informationisbeautiful.net/visualizations/million-lines-of-code/.

[6] J. García and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[7] M. Dibaei, X. Zheng, K. Jiang et al., "An overview of attacks and defences on intelligent connected vehicles," 2019, https://arxiv.org/abs/1907.07455.

[8] A. Chattopadhyay, K. Lam, and Y. Tavva, "Autonomous vehicle: security by design," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 11, pp. 7015–7029, 2021.

[9] W. Li, L. Bao, Y. Li, H. Si, and Y. Li, "Assessing the transition to low-carbon urban transport: a global comparison," *Resources, Conservation and Recycling*, vol. 180, Article ID 106179, 2022.

[10] M. Wiering and M. van Otterlo, *Reinforcement Learning*, Springer, Berlin, Germany, 2012.

[11] D. Silver, A. Huang, C. J. Maddison et al., "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[12] J. Duan, S. Eben Li, Y. Guan, Q. Sun, and B. Cheng, "Hierarchical reinforcement learning for self-driving decision-making without reliance on labelled driving data," *IET Intelligent Transport Systems*, vol. 14, no. 5, pp. 297–305, 2020.

[13] G. Li, Y. Yang, S. Li, X. Qu, N. Lyu, and S. E. Li, "Decision making of autonomous vehicles in lane change scenarios: deep reinforcement learning approaches with risk awareness," *Transportation Research Part C: Emerging Technologies*, vol. 134, 2022.

[14] L. Wang, W. Ma, L. Wang, Y. Ren, and C. Yu, "Enabling in-depot automated routing and recharging scheduling for automated electric bus transit systems," *Journal of Advanced Transportation*, vol. 2021, Article ID 5531063, 15 pages, 2021.

[15] M. Cheng, C. Zhang, H. Jin, Z. Wang, and X. Yang, "Adaptive coordinated variable speed limit between highway mainline and on-ramp with deep reinforcement learning," *Journal of Advanced Transportation*, vol. 2022, Article ID 2435643, 16 pages, 2022.

[16] Z. Ma, T. Cui, W. Deng, F. Jiang, and L. Zhang, "Adaptive optimization of traffic signal timing via deep reinforcement learning," *Journal of Advanced Transportation*, vol. 2021, Article ID 6616702, 14 pages, 2021.

[17] L. Zheng, B. Wu, and P. J. Jin, "A reinforcement learning based traffic control strategy in a macroscopic fundamental diagram region," *Journal of Advanced Transportation*, vol. 2022, Article ID 5681234, 12 pages, 2022.

[18] L. Elmoiz Alatabani, E. Sayed Ali, R. A. Mokhtar, R. A. Saeed, H. Alhumyani, and M. Kamrul Hasan, "Deep and reinforcement learning technologies on internet of vehicle (IoV) applications: current issues and future trends," *Journal of Advanced Transportation*, vol. 2022, Article ID 1947886, 16 pages, 2022.

[19] S. Wang, S. K. J. Chang, and S. Fallah, "Autonomous bus fleet control using multiagent reinforcement learning," *Journal of Advanced Transportation*, vol. 2021, Article ID 6654254, 14 pages, 2021.

[20] T. Zhu, X. Li, W. Fan, C. Wang, H. Liu, and R. Zhao, "Trajectory optimization of CAVs in freeway work zone considering car-following behaviors using online multiagent reinforcement learning," *Journal of Advanced Transportation*, vol. 2021, Article ID 9805560, 17 pages, 2021.

[21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, https://arxiv.org/abs/1707.06347.

[22] Minisry of Transport of the People's Republic of China, *Technical Standard of Highway Engineering*, China Communications Press Co. Ltd, Beijing, China, 2014.

[23] Minisry of Transport of the People's Republic of China, *Design Specification for Highway Alignment*, China Communications Press Co Ltd, Beijing, China, 2018.

[24] M. Kaspar, J. D. M. Osorio, and J. Bock, "Sim2Real transfer for reinforcement learning without dynamics randomization," in

Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4383–4388, IEEE, Las Vegas, NV, USA, January 2020.

[25] OpenAI, "CarRacing-v0," 2022, https://github.com/AGiannoutsos/car_racer_gym.

[26] X. Ma, "Reinforcement learning for gym CarRacing-v0 with PyTorch," 2022, https://github.com/xtma/pytorch_car_caring.

[27] D. P. Kingma and J. L. Ba, "Adam: a method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations*, San Diego, CA, USA, December, 2015.