Vehicles Control: Collision Avoidance using Federated Deep Reinforcement Learning

Badr Ben Elallid¹, Amine Abouaomar², Nabil Benamar^{1,2}, and Abdellatif Kobbane³

¹ University Moulay Ismail, Meknes, Morocco.

² Al Akhawayn University in Ifrane, Morocco.

³ ENSIAS, Mohammed V University in Rabat, Morocco.

Abstract-In the face of growing urban populations and the escalating number of vehicles on the roads, managing transportation efficiently and ensuring safety have become critical challenges. To tackle these issues, the development of intelligent control systems for vehicles is paramount. This paper presents a comprehensive study on vehicle control for collision avoidance, leveraging the power of Federated Deep Reinforcement Learning (FDRL) techniques. Our main goal is to minimize travel delays and enhance the average speed of vehicles while prioritizing safety and preserving data privacy. To accomplish this, we conducted a comparative analysis between the local model, Deep Deterministic Policy Gradient (DDPG), and the global model, Federated Deep Deterministic Policy Gradient (FDDPG), to determine their effectiveness in optimizing vehicle control for collision avoidance. The results obtained indicate that the FDDPG algorithm outperforms DDPG in terms of effectively controlling vehicles and preventing collisions. Significantly, the FDDPGbased algorithm demonstrates substantial reductions in travel delays and notable improvements in average speed compared to the DDPG algorithm.

Index Terms—Autonomous Vehicles, Federated Deep Reinforcement Learning, Vehicle Control, Multi agent reinforcement learning.

I. INTRODUCTION

The proliferation of vehicles on the urban roads led to increasing traffic congestion, longer travel times, and a higher number of road accidents [1], [2]. Legacy traffic management systems rely often on pre-determined rules, which may not be sufficient to tackle the continuous growing complexity and unpredictability of traffic scenarios. Recent work on machine learning opened new paths for developing intelligent vehicle control systems capable of adapting to the complex dynamic environments and making real-time decisions [3], [4]. Deep Reinforcement Learning (DRL), as a subfield of machine learning, has demonstrated a high efficiency in solving complex control problems through enabling agents to acquire optimal policies by interacting with their environment [5]–[7].

DRL combines the power of deep neural networks with reinforcement learning algorithms, enabling autonomous agents to learn optimal policies through trial and error in complex environments. By interacting with their surroundings, these agents can optimize their actions to achieve specific goals, such as minimizing travel time or avoiding collisions. In the realm of vehicle control, DRL has the potential to revolutionize how vehicles navigate through traffic, make driving decisions, and respond to unforeseen situations. By leveraging DRL techniques, we can develop advanced control systems that can autonomously adapt to dynamic traffic conditions, optimize driving behavior, and ultimately improve traffic efficiency and safety [8], [9].

In the context of vehicle control, DRL can be leveraged to optimize driving decisions in order to avoid collision, route planning, and traffic efficiency. we aim to go beyond the conventional use of DRL and explore the potential of Federated Deep Reinforcement Learning (FDRL) techniques, specifically the Federated Deep Deterministic Policy Gradient (FDDPG), for vehicle control in order to enhance collision avoidance and overall traffic management. One key aspect of our approach is the focus on privacy-preserving learning. We recognize the importance of keeping individual vehicle data private while still benefiting from the collective learning of other vehicles in their traffic environment. By utilizing FDRL techniques, we can achieve this objective, allowing vehicles to learn from each other's experiences without directly sharing sensitive data.

We provide also a comparison of local DDPG and global FDDPG models performance in terms of the travel delay and the average speed, also we assess their suitability for practical implementation in real-world traffic scenarios. Through extensive experimentation, we employed a widely-used traffic simulation frameworks, namely, Veins and SUMO, to evaluate the performance of the FDDPG algorithm for vehicle control and collision avoidance. We finally provided an analysis of our study to emphasis on the potential benefits and challenges associated with the adoption of FDRL techniques for vehicle control.

The contributions of this paper can be summararized as follows,

- We provided a comparison of the DDPG and FDDPG algorithms, then we evaluate their performance in terms of travel delay reduction and average speed improvement under different traffic scenarios.
- We employed traffic simulation frameworks, namely, Veins and SUMO, to create a realistic environment for simulating vehicular networks and evaluating the effectiveness of the compared algorithms in real-world traffic setup.
- We demonstrated the superiority of the FDDPG algorithm over the DDPG algorithm in controlling vehicles for

collision avoidance and to improve traffic efficiency and safety.

The remainder of this paper is as follows. Section II discusses different related works. Section III presents the System model and different entities composing the considered model. Section IV discusses the proposed solutions and the comparison between FDDPG and DDPG setups. Section V is dedicated to discussing the finding. Section VI conclude the paper.

II. RELATED WORKS

The work of [10] developed a DRL-based collision avoidance method for unmanned surface vehicles (USVs) that focuses on the decision-making stage. The aim of this work is to determine if an avoidance maneuver is necessary and. if so, the direction of the maneuver. The authors proposed a neural network architecture and a semi-Markov decision process model for the USV collision avoidance. The DRL network is trained through a number of simulations and then implemented in experiments and simulations to evaluate its situation recognition and collision avoidance performance. The work in [11] aimed at exploring the potential of DRL into solving collision avoidance problems in unknown and compact environments. The proposed approach is compared to different traditional methods, such as potential field-based methods and the dynamic window approach. In [12], authors proposed a decentralized DRL framework for collision avoidance, where each agent independently makes decisions without communicating with other agents. The proposed approach enables mobile robot agents to learn efficient obstacle avoidance and navigation towards a targeted point in the environment with different dynamic obstacles. Soft actor-critic algorithm is employed to train the agents on obstacle avoidance policies in dynamic environments. In [13], authors focused on enhancing the quality and safety of autonomous driving control by utilizing DRL as an alternative to traditional rule-based control strategies. Authors employed DDPG and Recurent DPG algorithms, combined with Convolutional Neural Networks (CNN), to enable autonomous driving control for self-driving cars. In this paper, we aim at comparing most investigated methods from the literature and provide a comprehensive summary on pros and cons of each of the methods.

III. A BRIEF OVERVIEW OF FEDERATED DEEP REINFORCEMENT LEARNING

DRL has emerged as a powerful technique in the field of Autonomous Vehicles (AVs) and has the potential to significantly improve the safety of autonomous driving. DRL is a lightweight technology that enables quick decisions and actions in real-time. DRL is based on the interaction between an agent and its environment, where the agent performs various actions and receives a reward for each performed action [14]. At each step t, the agent observes a state S_t in the environment, chooses an action a_t , gets a reward r_{t+1} , and the environment transitions to the next state S_{t+1} . One way to define Reinforcement learning is the Markov Decision Process (MDP), which includes several terms (S, A, T, R) with the following conditions:

- 1) S is a set of all possible states of an environment ($s \in S$)
- 2) \mathcal{A} is a set of actions that an agent can take $(a \in \mathcal{A})$
- T: S×A×S → [0,1] is the transition state that provides the probability associated with executing an action in a state and transitioning to a new state.
- *R* : *S*×*A*×*S* → ℝis the reward that provides the penalty for taking an action in a state and transitioning to a new state.

A probability of taking action a_t in state s_t , so-called policy and denote it by π . This probability is expressed as follow:

$$\pi(a_t|s_t) = P[A = a_t|S = s_t]$$

In contrast to (MDP), DRL is a model-free algorithm that does not require any probability modeling of the environment.

The DRL algorithm is suitable for use in the field of autonomous driving due to its capacity to manage new and unobserved environments. The DRL requires two main steps: 1) The agent tries to explore its environment by taking random actions; 2) The agent exploits prior knowledge by taking advantage of the exploration phase and follows an optimal policy π_* to achieve the best action a_t by maximizing the sum of subsequent rewards, identified as follows:

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots + \gamma^T r_{t+T} = \sum_{k=0}^T \gamma^k r_{t+k}$$

Where $\gamma \in [0, 1]$ represents the discount factor that penalizes future rewards, and T represents the end time of episode. DDPG is an off-policy, model-free, actor-critic algorithm based on the Deterministic Policy Gradient (DPG) theorem [15].

DDPG combines DQN and policy gradients to learn deterministic policies. DDPG learns Q-value and policy based on off-policy data and the Bellman equation 1:

$$Q_*(s_t, a_t) = E\left[r_t + \gamma \max_{a_{t+1}} Q_*(s_{t+1}, a_{t+1})\right]$$
(1)

Federated Deep Reinforcement Learning (FDRL) combines Federated Learning and Deep Reinforcement Learning to enable collaborative and privacy-preserving training of reinforcement learning models. Multiple devices or entities, such as vehicles or IoT devices, train their own local models without sharing raw data. The central server aggregates model updates, creating a global model that captures collective knowledge. FDRL benefits from collaborative learning while maintaining data privacy and security. It has the potential to transform applications like autonomous vehicles, allowing vehicles to learn from each other without sharing sensitive data.

In this paper, we propose a technique for controlling autonomous vehicles (AVs) in traffic with the goal of reaching their destination safely and collision-free using Federated DDPG algorithm.



Fig. 1: The FDRL framework focuses on collision avoidance (RL) tasks in (AVs). In this framework, wireless communication is established between all ten agents and the federated learning (FL) server. Each agent operates autonomously within its own environment. Each round, the FL server collects and aggregates the RL models from all agents, generating a global model. This global model undergoes asynchronous updates from various RL agents, fostering collaborative learning within the system.

IV. FEDERATED DEEP REINFORCEMENT LEARNING

A. RL Problem Formulation

- State space: Our simulation consists of various features of the vehicle received by the environment in each step. Specifically, the state at time step t is represented by a vector $S_t = (PosX_t, PosY_t, Speed_t, \theta_t, A_t, Dest)$, where $PosX_t$ and $PosY_t$ denote the x and y positions of the vehicle, θ_t is its orientation, $Speed_t$ and A_t are its velocity and acceleration, respectively. Additionally, Dest represents the distance to the destination, which is calculated as the Euclidean distance between the current position of the vehicle Pos_{AV} and the end position Pos_{end} , i.e., $||Pos_{AV} - Pos_{end}||$. To obtain these features, we use TraCi, which is a tool provided by Sumo Simulator as shown in Figure 1.
- Action space: is a control command that is sent to our vehicle in the environment. Since DDPG is a continuous DRL algorithm, it is necessary to use continuous action. In our case, we use acceleration or deceleration as our control command that we send to the vehicle in the Sumo Simulator.
- State Transition: is the probability of executing action a_t in state s_t at time t and transitioning new state s_{t+1} at time t+1: $P = P(s_{t+1}|s_t, a_t)$
- Reward function: is a function that generates reward rt while executing action at. In our case, the vehicle should reach its destination as quickly as possible without colliding with other vehicles on its route, reducing time spent waiting at traffic lights or if some vehicles are braking. Let V_{Speed}, Pos, Pos_{dest}, Traffic_{braking}, Traffic_{waiting} denote the speed of vehicles, the vehicle's position, the destination's position, and our vehicle is braking or waiting in traffic light, respectively. If the vehicle's speed is not zero, then a penalty of 0.04 is assessed. If the simulator detects a crash, a penalty of -5 is obtained. To reduce waiting time, the reward is -0.05 if our vehicle brakes and waits at a traffic light. The reward is -0.025 if our vehicle is braking or

waiting at a traffic light; otherwise, the penalty of -0.02 is attributed to the agent. Furthermore, the agents reward of 5 is affected if the vehicle reaches its target destination.

$$reward = \begin{cases} -10 & \text{if there is collision} \\ 0.04 & \text{if } V_{Speed} \neq 0 \\ -0.05 & Traffic_{braking} \text{ and } Traffic_{waiting} \\ 0.025 & Traffic_{braking} \text{ or } Traffic_{waiting} \\ 0.05 & \text{not } Traffic_{braking} \\ 0.05 & \text{not } Traffic_{waiting} \\ 10 & \text{To reach destination} \\ -0.02 & \text{by default} \end{cases}$$

B. Federated DDPG model

Our DDPG model consists of two networks: the Actor and the Critic. The Critic predicts the action value based on the state and the value obtained by the Actor. The Actor network takes the state s_t as input and predicts the action taken by the agent. Ornstein-Uhlenbeck noise is added to this action to encourage exploration, as part of the policy $(a_t = \mu(s_t | \theta^{\mu}) + N_t)$. The agent's experience, including the state s_t , action taken a_t , reward received r_t , and next state s_{t+1} , is stored as a tuple in the replay buffer. We randomly select a batch of tuples from the replay buffer, and the Critic network takes the state s_t and action taken a_t to predict the Q-value. To update the Q-value using Equation 1, we need a second pass through both the Actor and the Critic networks, which are referred to as the target Actor and the target Critic networks, respectively. Next, we compute the loss between the predicted Q-value and the value obtained from Equation 1. Finally, we update the weights of the Actor and Critic networks using the following formulas: $\theta' \leftarrow \tau \theta + (1 - \tau) \theta'$ and $\phi' \leftarrow \tau \phi + (1 - \tau) \phi'$. Here, θ and ϕ are the weights of the Actor and Critic networks, respectively, and τ is a



Fig. 2: Illustration of Meknes city map which used like our environment to train FDDPG models. Figure (a) illustrates the hole map used in our training, while Figure (b) shows an example of traffic in our map involving other vehicles.

hyperparameter that controls the rate of updating the target networks.

The training process of Federated DDPG consists of five rounds. In each round, ten agents representing ten vehicles are trained in their respective traffic environments. After training, the weights of each agent are sent to an aggregation process to calculate the average of weights using this formula.

$$w_{avg} = \frac{\sum_{i=1}^{N} n_i \cdot w_i}{\sum_{i=1}^{N} n_i}$$

Where w_i represents the weights of agent *i*, *N* represents the total number of agents, and n_i is is the number of episodes contributed by agent i

In order to illustrate more how the FDDPG works, we provide the following pseudo-code

V. SIMULATION RESULTS

A. Setup

Nowadays, testing autonomous vehicle approaches in hyperrealistic virtual environments is the most important part of discovering comfortable AV systems. According to the complexity of the scenarios of autonomous driving, it is necessary to simulate and validate the proposed approaches in simulators close to the real world in order to ensure the efficiency of these models. For these reasons, we evaluate our model in Sumo Simulator [16], which is considered the most powerful simulator for traffic simulation since it allows export of maps from the real world using Open Street Maps. Open Street Maps allows us to import a large road network from maps, which allows us to simulate our model in real-world conditions.

Algorithm 1 Federated Deep Deterministic Policy Gradient (FDDPG)

- 1: Initialize global critic network $Q_{\phi}(s, a)$ with random weights ϕ
- 2: Initialize global actor network $\mu_{\theta}(s)$ with random weights θ
- 3: Initialize target critic network $Q'_{\phi}(s, a)$ with weights $\phi' \leftarrow$ ϕ
- 4: Initialize target actor network $\mu'_{\theta}(s)$ with weights $\theta' \leftarrow \theta$
- 5: for each federated device i in parallel do
- Initialize local critic network $Q_{\phi_i}(s, a)$ with weights 6: $\phi_i \leftarrow \phi$
- Initialize local actor network $\mu_{\theta_i}(s)$ with weights $\theta_i \leftarrow$ 7: θ
- 8: Initialize local replay buffer \mathcal{D}_i
- 9: end for

14:

16:

- 10: **for** each iteration **do**
- for each federated agent *i* in parallel do 11:
- Sample mini-batch of experiences from local re-12: play buffer \mathcal{D}_i
- Update local critic network $Q_{\phi_i}(s, a)$ by minimiz-13: ing the loss function:

$$L(\phi_i) = \frac{1}{N} \sum_{j=1}^{N} (Q_{\phi_i}(s_j, a_j) - y_j)^2$$

Update local actor network $\mu_{\theta_i}(s)$ using the policy 15: gradient:

$$\begin{array}{c} \nabla_{\theta_i} J(\theta_i) \\ \frac{1}{N} \sum_{j=1}^{N} \nabla_a Q_{\phi_i}(s_j, a_j) \nabla_{\theta_i} \mu_{\theta_i}(s_j) \\ \text{Update local target networks:} \end{array}$$

 \approx

17: $\phi'_i \leftarrow \tau \phi_i + (1 - \tau) \phi'_i$ $\theta'_i \leftarrow \tau \theta_i + (1 - \tau) \theta'_i$ 18

$$\begin{array}{ccc} 18: & \phi_i \leftarrow \tau \phi_i \\ 19: & \theta'_i \leftarrow \tau \theta_i + \end{array}$$

$$\theta_i \leftarrow \tau \theta_i + (1 +)$$

20: end for

- Aggregate local critic network weights $\{\phi_i\}$ and actor 21: network weights $\{\theta_i\}$ at the global server
- 22: Update global critic network $Q_{\phi}(s, a)$ with aggregated weights
- Update global actor network $\mu_{\theta}(s)$ with aggregated 23: weights
- Update global target networks: 24:

25:
$$\phi' \leftarrow \tau \phi + (1 - \tau) \phi$$

 $\theta' \leftarrow \tau \theta + (1 - \tau) \theta'$ 26:

27: end for

The PythonAPI is provided by SUMO and manages various scenarios. Further, we combine Sumo with Gym, which is an open-source Python library created by OpenAI [17]. It was initially intended to be used for the creation and evaluation of reinforcement learning algorithms. The gym facilitates communication between the DRL algorithm and the environment. For this purpose, our model aims to benefit from the power of OpenAI and Sumo simulation to accelerate the training of our model and get the best results. On the other hand, we use TraCi [18] to control and communicate with vehicles in our environment.

We imported a map of the city of Meknes, Morocco using

Open Street Map, as shown in Figure 2, to simulate 4868 vehicles navigating randomly across the map. The speed limit and the collision sensitivity were set to 20 m/s and 2 respectively.

The training episodes terminate under one of the following conditions: 1) the ego vehicle collides with other vehicles, 2) the ego vehicle reaches its destination, or 3) the number of steps in the episode exceeds the maximum steps (900). We have designed our architecture with two fully connected hidden layers in both the actor and critic networks. The actor network consists of 400 neurons in the first hidden layer and 300 neurons in the second hidden layer. Similarly, the critic network also has 400 neurons in the first hidden layer and 300 neurons in the second hidden layer. The input layer is of size 6, representing the shape of the state S_t , and the output layer size is 1, predicting the action. To prevent the issue of vanishing gradients during backpropagation, we used the rectified linear unit as our activation function. We implemented our proposed model in Python, using the PyTorch library to construct the DDPG architecture. Training was conducted on a machine equipped with an Intel Core i7-11800H (8-Core, 2.3GHz) processor, 16GB of RAM, and a single NVIDIA RTX 3050 GPU. We employed the Adam optimizer, and additional parameters are specified in Table I.

TABLE I: Parameters of DDPG model

Parameter	Value
Actor/critic learning rate	5×10^{-4}
Episodes	500 for each agent
Actor/critic batch size	64
γ	0.99
Replay Memory Size	50000

B. Result



Fig. 3: Average reward of each round during episodes

The average rewards of the FDDPG model are shown in Figure 3. In the initial round, the curve exhibits fluctuations, reaching both high and low values. This variability arises from variations in learning among the agents, with some learning faster than others. However, starting from the second round, we observe a consistent increase in the average reward, eventually stabilizing. Notably, in the final round, all agents achieve a high and stable score, indicating the rapid learning capability of our model.

In our evaluation, we aim to determine the effectiveness of two models in controlling vehicles within their traffic scenarios. We analyze multiple metrics including collisions, travel delay, and average vehicle speed to assess their performance. To ensure accurate and reliable results, we conduct 20 simulation episodes and measure the aforementioned metrics.

During the simulation, the AVs are controlled by either a trained actor network for the local DDPG model or the global FDDPG model. The objective is to guide the AVs towards their respective target destinations while minimizing travel delay and avoiding collisions. To facilitate a comprehensive evaluation, we specifically chose four AVs and conducted performance comparisons between the local model DDPG and the global model FDDPG.

- Collision: We assessed the performance of an Autonomous Vehicle (AV) using Traci and FDDPG controllers through simulation. Specifically, we focused on measuring the collision occurrences at various destination positions. We chose five destination positions located at Euclidean distances of 10m, 20m, 52m, 107m, and 207m from the starting point to the endpoint. Our findings indicate that when the AV was controlled by the FDDPG controller, it effectively avoided collisions with other vehicles. Conversely, when the AV was controlled by DDPG, collisions with other vehicles were observed in every episode.
- **Travel delay** : we evaluated the impact of local and global models on reducing travel delay for four AVs across five different distances. Figure 4 provides a illustrated of travel delay for AVs. Our findings revealed that both the local and global models contributed to a significant reduction in travel delay, allowing the AVs to reach their destinations more efficiently. Notably, the global model, specifically the FDDPG model, outperformed the local DDPG model in terms of facilitating faster achievement of the AVs' destinations.
- Average speed: To evaluate the speed performance of the autonomous vehicles (AVs), we analyzed their average speed across five different distances. The average speed was computed by summing the speeds of the AVs over a single episode and dividing it by the number of steps taken to reach their destinations. Figure 5 provides a visual representation of the results.

Our analysis revealed that the average speed of the AVs, controlled by both the DDPG and FDDPG algorithms, increased as the distance between the start and end points increased. This suggests that the AVs learned to accelerate their speeds to reach their destinations more quickly. Additionally, we observed that the FDDPG algorithm outperformed the DDPG algorithm in terms of achieving higher average speeds for the four AVs.



Fig. 4: Travel delay of four vehicles in different distances



Fig. 5: The average speed of four vehicles over distances

VI. CONCLUSION

In this study, our aim was to analyze vehicle control for collision avoidance using Federated Deep Reinforcement Learning (FDRL) techniques, while simultaneously minimizing travel delays, enhancing average vehicle speed, and ensuring safety. To achieve this, we compared the performance of local and global models, namely DDPG and FDDPG, to determine the most effective approach for optimizing vehicle control in collision avoidance scenarios. Through simulations, we observed that the FDDPG model outperformed the DDPG algorithm in terms of controlling vehicles and preventing collisions, all while maintaining data privacy. The FDDPG-based approach exhibited a noticeable reduction in travel delays and an increase in average speed when compared to the DDPG algorithm. These results highlight the potential of FDDPG for practical implementation in intelligent transportation systems. Furthermore, to create realistic environments for evaluating the performance of the deep reinforcement learning algorithms in various traffic setups, we utilized traffic simulation frameworks such as Veins and SUMO.

REFERENCES

- B. B. Elallid, N. Benamar, A. S. Hafid, T. Rachidi, and N. Mrani, "A comprehensive survey on the application of deep and reinforcement learning approaches in autonomous driving," *Journal of King Saud University-Computer and Information Sciences*, 2022.
- [2] M. E. Hanjri, H. Kabbaj, A. Kobbane, and A. Abouaomar, "Federated learning for water consumption forecasting in smart cities," *arXiv* preprint arXiv:2301.13036, 2023.
- [3] B. B. Elallid, M. Bagaa, N. Benamar, and N. Mrani, "A reinforcement learning based approach for controlling autonomous vehicles in complex scenarios," in 2023 International Wireless Communications and Mobile Computing (IWCMC), pp. 1358–1364, IEEE, 2023.

- [4] B. B. Elallid, S. E. Hamdani, N. Benamar, and N. Mrani, "Deep learningbased modeling of pedestrian perception and decision-making in refuge island for autonomous driving," in *Computational Intelligence in Recent Communication Networks*, pp. 135–146, Springer, 2022.
- [5] S. Feng, H. Sun, X. Yan, H. Zhu, Z. Zou, S. Shen, and H. X. Liu, "Dense reinforcement learning for safety validation of autonomous vehicles," *Nature*, vol. 615, no. 7953, pp. 620–627, 2023.
- [6] B. B. Elallid, N. Benamar, N. Mrani, and T. Rachidi, "Dqn-based reinforcement learning for vehicle control of autonomous vehicles interacting with pedestrians," in 2022 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), pp. 489–493, IEEE, 2022.
- [7] H. Kabbaj, M. El Hanjri, A. Kobbane, R. El-Azouzi, and Abouaomar, "Distfl: An enhanced fl approach for non trusted setting in water distribution networks,"
- [8] A. Abouaomar, Z. Mlika, A. Filali, S. Cherkaoui, and A. Kobbane, "A deep reinforcement learning approach for service migration in mecenabled vehicular networks," in 2021 IEEE 46th Conference on Local Computer Networks (LCN), pp. 273–280, IEEE, 2021.
- [9] A. Abouaomar, A. Taik, A. Filali, and S. Cherkaoui, "Federated deep reinforcement learning for open ran slicing in 6g networks," *IEEE Communications Magazine*, 2022.
- [10] J. Woo and N. Kim, "Collision avoidance for an unmanned surface vehicle using deep reinforcement learning," *Ocean Engineering*, vol. 199, p. 107001, 2020.
- [11] S. Feng, B. Sebastian, and P. Ben-Tzvi, "A collision avoidance method based on deep reinforcement learning," *Robotics*, vol. 10, no. 2, p. 73, 2021.
- [12] M. Raibail, A. H. A. Rahman, G. J. AL-Anizy, M. F. Nasrudin, M. S. M. Nadzir, N. M. R. Noraini, and T. S. Yee, "Decentralized multi-robot collision avoidance: A systematic review from 2015 to 2021," *Symmetry*, vol. 14, no. 3, p. 610, 2022.
- [13] C.-C. Chang, J. Tsai, J.-H. Lin, and Y.-M. Ooi, "Autonomous driving control using the ddpg and rdpg algorithms," *Applied Sciences*, vol. 11, no. 22, p. 10659, 2021.
- [14] P. Palanisamy, Hands-On Intelligent Agents with OpenAI Gym: Your guide to developing AI agents using deep reinforcement learning. Packt Publishing Ltd, 2018.
- [15] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference* on machine learning, pp. 387–395, Pmlr, 2014.
- [16] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd,

R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*, IEEE, 2018.

- [17] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.
- [18] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J.-P. Hubaux, "Traci: an interface for coupling road traffic and network simulators," in *Proceedings of the 11th communications and networking simulation symposium*, pp. 155–163, 2008.