Int. j. inf. tecnol. (October 2023) 15(7):3541–3553
https://doi.org/10.1007/s41870-023-01412-6

ORIGINAL RESEARCH

Deep reinforcement learning for autonomous vehicles: lane keep and overtaking scenarios with collision avoidance

S. H. Ashwin[1] · **Rashmi Naveen Raj**[1]

Received: 16 November 2022 / Accepted: 15 May 2023 / Published online: 2 September 2023

Abstract Numerous accidents and fatalities occur every year across the world as a result of the reckless driving of drivers and the ever-increasing number of vehicles on the road. Due to these factors, autonomous cars have attracted enormous attention as a potentially game-changing technology to address a number of persistent problems in the transportation industry. Autonomous vehicles need to be modeled as intelligent agents with the capacity to observe, and perceive the complex and dynamic environment on the road, and decide an action with the highest priority to the lives of people in every scenarios. The proposed deep deterministic policy gradient-based sequential decision algorithm models the autonomous vehicle as a learning agent and trains it to drive on a lane, overtake a static and a moving vehicle, and avoid collisions with obstacles on the front and right side. The proposed work is simulated using a TORC simulator and has shown the expected performance under the above-said scenarios.

Keywords Autonomous vehicles · Reinforcement learning · Smart city · Deep deterministic policy gradient · Obstacle detection

✉ Rashmi Naveen Raj
rashmi.naveen@manipal.edu

S. H. Ashwin
ashwin.h1@learner.manipal.edu

[1] Department of Information and Communication Technology, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, India

1 Introduction

Autonomous vehicles utilize multiple sensors such as LiDARs, RADARs, GPS-GNSS, cameras, etc. to perceive their surroundings and move with scant or no human interaction [1]. In the present world, many tasks are being automated to provide humans with more convenience and safety. There can be scenarios where the drivers are not in a good state to drive, and it may lead to accidents. Instead, if the task of driving is given to an adequately trained machine, it will perform its task with maximum efficiency every single time. The cars became autonomous first in the 1920s and were called "Phantom Autos" [2]. They were called so because they were remote-controlled. Later in the 1980s self-managed autonomous vehicles were introduced by Mercedes, which were not fully automatic. Many automobile companies like Tesla, Waymo, Baidu, Chevrolet Cruise, etc are working towards the development and commercialization of driverless self-driving vehicles. The advancements in automobiles, sensor technology, image and video processing, information technology, and the increasing interest in designing autonomous vehicles by many young researchers will definitely result in low-cost, reliable, and efficient safe driverless vehicles on the road in the near future.

The Society of Autonomous Engineers (SAE) has defined five different levels of automation in SAE_J3016 as shown in Table 1 [3] [4]. Levels (0 to 2) still need drivers' support and levels (3 to 5) are defined to have driverless automated features. The number of sensors, controlling functions, and computing cost increases with each level, and also the cost of the vehicle.

Figure 1 depicts the layer-wise components and functions of an autonomous vehicle system [1, 5]. At layer 1, the autonomous vehicles are equipped with various multimodal sensors to detect surrounding objects, object size

Table 1 Automation levels

Levels	Definition
Level 0	No or Zero automation
Level 1	Driver assistance for one or two features (steering or braking)
Level 2	Partial automation with continuous monitoring from the driver
Level 3	Advanced partial automation/ driver controls the vehicle only when there is a notice to intervene
Level 4	High automation with human driver
Level 5	Full automation and no human driver is required

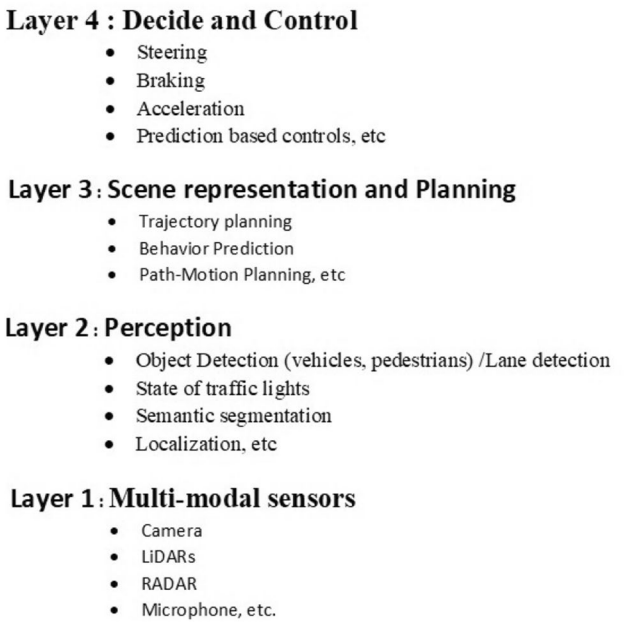

Fig. 1 Layered architecture of autonomous driving system

or depth, road marking, and distance of the obstacles [1, 6]. The localization devices like GPS are used to get the location. The cameras are used to detect depthless objects like traffic instructions [7]. Microphones may be used to identify the sound of an ambulance or fire engine and so on. Layer 2 processes the information received from different sources of layer 1 to create a map of the surrounding environment. Thus, layer 2 is referred to as the perception layer as it recognizes: the distance of the obstacle from the ego vehicle, the curve ahead, the lane, and so on [4, 7]. Thus, the perception system needs to be precise and robust [6]. Even a small error in this system can give rise to deadly accidents. Next is the planning layer which plans for the next action based on the understanding derived from the previous layer. The last layer is the control layer which generates the control signals based on the decision taken at the planning layer.

Researchers and industrialists have explored various Machine Learning (ML) models to automate driving strategies. In supervised learning algorithms, the model is trained beforehand using labeled data to perform a certain task [8] whereas in unsupervised learning the model is forced to learn from unlabelled data. Labeled data is a collection of information with one or more labels. These labels are useful tags that help ML models to quickly interpret the data. Labeled data are the cornerstone of supervised learning [9] and require a large amount of labeled video data for training an autonomous vehicle. But, generating or creating a data set is a tedious and costly process. The third category is Reinforcement Learning (RL) which does not need a dataset and an agent improves its performance continuously through constant interaction with its surrounding environment.

For the process of autonomous driving which is a sequential decision problem, RL is the optimal choice as the vehicle needs to actively interact with its environment to drive safely. It is impossible to train the vehicle for all possible scenarios beforehand thus ruling out supervised and unsupervised learning approaches. There are a few challenges to, using RL though, such as bridging the gap between simulation and reality, sample efficiency, etc. Research in applying RL for autonomous driving requires tremendous effort from academicians, researchers, and automotive industry experts from various fields to bring this idea into reality. The contributions of the proposed work are:

- A Deep RL (DRL) -based sequential decision model is proposed to learn safe driving strategies.
- The proposed algorithm trains the agent to avoid or alleviate frontal and side collisions with minimum risk.
- The proposed algorithm is trained and verified using The Open Racing Car Simulator (TORCS) and outperforms under various scenarios.

The paper is organized as follows: The various RL algorithms are discussed briefly in Sect. 2. Section 3 presents the state-of-the-art research in RL-based autonomous driving. Section 4.1 explains the proposed methodology and Sect. 4.3 presents the implementation of the proposed work using TORCS. Finally, the results obtained are discussed in Sects. 5 and 6 concludes the research work and future directions are presented in Sect. 7.

2 Preliminaries

In the RL model, an autonomous agent masters its performance through persistent interaction with its environment [10]. The reward function is a performance criterion that is used to evaluate the RL agent [11]. The general scenario of RL is shown in Fig. 2. At any time instance t, the agent in

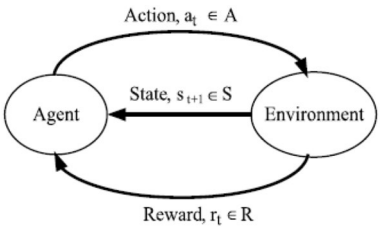

Fig. 2 General scenario of RL [11]

state s_t which belongs to the state space S, chooses an action a_t belonging to the action space A and receives a reward $r_t \epsilon R$ rom the environment based on the efficacy of its decision. The agent then moves to the next state s_{t+1} and proceeds to choose an action for this state. The cycle continues until the agent has learned the task completely or reached the final state. The agent's objective is to maximize the accumulative rewards obtained during the entire lifetime of the task. Eventually, by exploiting the knowledge learned, an agent can increase the lifetime reward. It expands its knowledge by trial and error method [11].

The prime challenge in RL is to balance exploration and exploitation. To enhance the rewards, an agent should exploit the gained knowledge by choosing actions that prove to be highly rewarding. In contrast, to ascertain such favourable actions, it has to make certain risky decisions. This may or may not result in higher rewards [12]. The strategy used by the agent to choose an action for a given state is known as a policy π. If a particular policy gives a maximum reward, such a policy is termed as an optimal policy π^*. RL can be used to solve many real-world problems but sometimes, there exists certain situations where conventional RL algorithms fail to provide desirable results. This is mainly due to the fact that the state spaces and action spaces involved in these problems are very highly dimensional e.g. camera images, infrared images, etc. and it is impossible to store all state-action pairs. To solve such problems, Deep Learning (DL) is used along with RL and is referred to as DRL [13]. In DRL, the policy is represented using a neural network. DRL is proven to be successful in many domains: game environments [14, 15], healthcare [16], communication networks [17], etc. without the need for a mathematical model or labeled data.

Deep Deterministic Policy Gradient (DDPG) and Deep Q-Network (DQN) are the two commonly used DRL algorithms. DDPG is an off-policy and model-free algorithm that deals with continuous state spaces and DQN deals with discrete state spaces. Instead of representing the policy as a probability distribution, DDPG uses gradient descent to create a policy that is deterministic in nature. The main advantage of DDPG compared to other stochastic policy algorithms is that it is simpler and values can be computed more efficiently. The conventional Q-learning algorithm calculates the Q values for state-action pairs and stores it in tabular form. However, for huge state and action spaces, it becomes infeasible to create a table of Q values. Therefore, in DQN, neural networks are used because the memory and computation required would be less compared to DQN [18]. A deep Q-learning function approximator is used to solve this problem. The DDPG algorithm is used in our proposed work and the same is illustrated in Fig. 3.

The agent is the self-driving vehicle which is in a state s and sends its state information to the actor. The actor decides the best possible action a and sends it back to the agent. After implementing the action, an agent gets a reward r (which can be anything depending upon how good the action was) and progresses to a new state s'. The entire transition comprising of (*state* s, *action* a, *reward* r, *new state* s') is stored in a replay buffer R. From this buffer, random sampling is done and a few transitions are stored in a mini-batch. The state s and action a are then sent to the critic to evaluate the action a taken by the actor. The critic generates a Q value for the state-action pair and this value shows how good the action was for that particular state. From the mini batch, the next state s' is sent to the target actor. The target actor is expected to give the action a' that is most suitable for the next state s'. This next state s' and next action a' is then sent to the target critic. The role of the target critic is to evaluate the action a' that was recommended by the target actor for the state s' and generate a Q value for (s', a') pair. Using the Q values for both (s', a') and (s, a) pairs, the loss function is calculated. Equation 1 [18] gives the loss function and is used to update the critic network. This helps the critic evaluate the actor's actions in a better way. Furthermore, the policy gradient is calculated using the Q value of (s, a) , which is then used to update the actor network. This helps the actor make better decisions.

$$L = \frac{1}{N}\Sigma_i(y_i - Q(s_i, a_i|\theta^Q))^2 \tag{1}$$

Target networks are not updated with calculated values. Instead, a soft update is carried out after the actor and critic networks are updated. The soft update involves updating the target network by a very small factor called τ with a typical value of 0.001. Once all these steps are carried out, the agent will have reached a new state, and the cycle continues. The actor uses a neural network to determine the desirable action, a whereas the critic employs a neural network to compute Q values.

3 Related works

Researchers have used different approaches like DDPG, DQN, imitation learning, transfer learning, etc. to build a

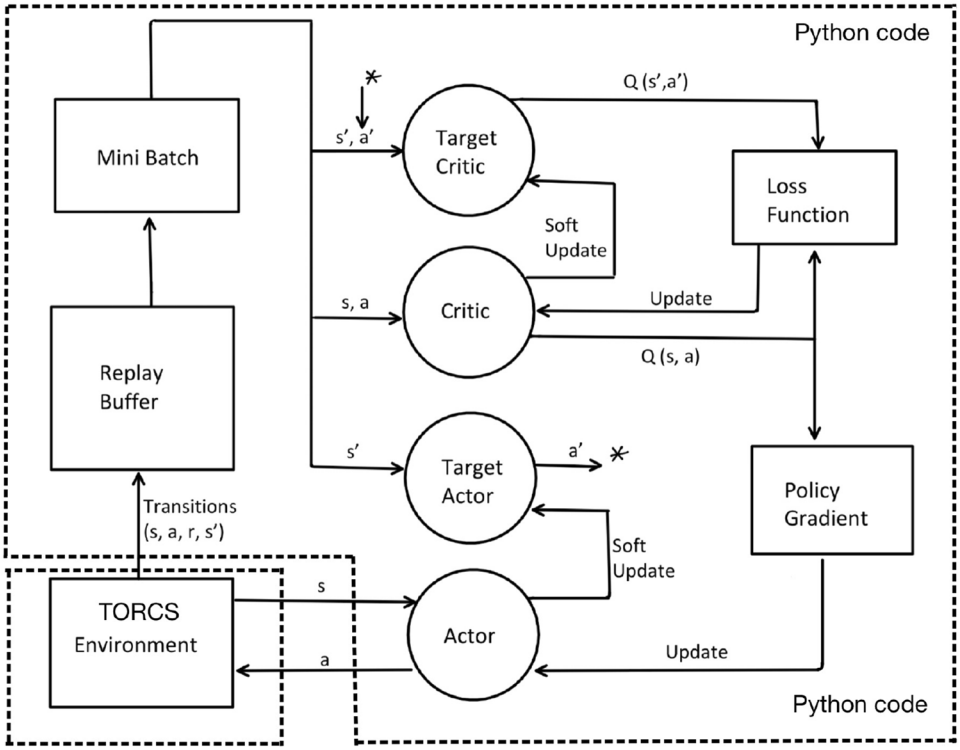

Fig. 3 Deep deterministic policy gradient block diagram

safe and efficient self-driving vehicles at various layers. An overview of the extant literature is provided in this section.

The work by Zhao et al. [19] aims to model the decision-making and interactions between various vehicles that run on highways. The authors have used Double DQN (DDQN) to train the host vehicle, and the proposed work is implemented using an open-source simulation platform called "SUMO - Simulation of Urban Mobility". The driving environment is created by having three driving lanes and randomly distributing 20 cars on the highway. The host continuously gauges the distance between itself and the impediment (which could be a moving vehicle) in front of it as it travels. It starts to apply the brake to prevent a collision if there is a drop in the distance between successive measurements. The speed of the host is altered accordingly by the algorithm. The work by Zhang et al. [20] focuses on employing DDQN to regulate vehicle speed. In order to train the RL agent and teach it human cognitive behaviour, authors have gathered factual driving data from actual human driving. DDQN is seen to increase the DQN's stability and dependability. According to the authors, the DDQN model scored significantly higher than the DQN model.

Chopra et al. [21] have aimed at steering the vehicle in its path with the help of the DQN algorithm. The model builds a Q-value approximator that regulates the car's steering using unprocessed images, sensor inputs, and estimated rewards. Due to the longer trainig time, there is potential for future use of imitation learning, which initially trains the model using labelled data before applying the RL algorithm to it. The survey by Elallid et al. [22] focuses on DL and RL-based approaches for the major functionalities of autonomous vehicles. This study examines research on DL/RL for autonomous cars from 2016 through 2021. The authors conduct a thorough comparison with regard to the aforementioned functionalities. One of the main challenges is the behavior of autonomous vehicles in different weather and lighting conditions.

Rasheed Hussain et al. [23] have focused on the outcomes achieved so far and the difficulties that researchers will face in the future. Over the past century, the car industry has made enormous advancements in the creation of dependable, safe, and efficient automobiles. Autonomous vehicles are becoming a reality due to enormous advancements in computer and communication technologies. The work by Zhu et al. [24] involves an agent learning to follow the lead vehicle that is being driven by a human in front of it. Only one sensor is utilized to determine the time for a collision, jerk in driving, etc. is the distance between the agent and the lead car. The reward function is developed by observing human driving data captured in real-time from the lead car and later combining it with driving-related features such as efficiency, comfort, and safety. Authors have used 'Next Generation Simulation' software for the implementation.

The research paper by Omeiza et al. [25] elucidates *explainable autonomous driving*. According to the authors, *Explainability* is an essential prerequisite for autonomous

vehicles. autonomous vehicles must be able to explicate what they see, do, and might do in the environments in which they interact. The reason for explanations, the specifications for explanations for autonomous vehicles, a review of previous work on explanations for autonomous vehicles, and finally a conceptual framework for explainability of autonomous vehicles are the four key issues covered in this study. Grigorescu et al. [26] have discussed the major open challenges in the field of autonomous driving. Also, authors indicate that DL and Artificial Intelligence (AI) will play a phenomenal role in overcoming these challenges. Among the major areas they addressed, functional safety, real time computing, and communication prove to be really challenging.

Yang et al. [27] elucidate about how AI is included into the development of the four key functions of autonomous driving: perception, localization, mapping, and decision-making. It explores the recent approaches to comprehend how AI can be made use of and what are the issues linked with their implementation. Based on the survey of current practices and advancement of the technologies, this paper further provides acumen into impending applications regarding the use of AI in juxtaposition with other emerging technologies. Cao et al. [28] discuss improving RL in a trustworthy manner (TiRL). Although RL algorithms have the capacity to continuously get better, it has been noted that there are times when they are unstable. The authors have created a framework for generating decisions that combines RL and rule-based algorithms in order to achieve the best results from both worlds. By doing this, their final framework may learn more reliably on its own in the future. Simulations are conducted with more than 42,000 kms of driving, calibrated with data from actual driving. Authors have demonstrated the superior performance of TiRL above all other arbitrary model-based policies.

A Raspberry Pi 3 Model-B microprocessor and Pi Cam Rev. 1.3 were used to construct vision-based outdoor navigation by Kumar et al. [29]. The front camera captures a picture of the front lane, which is then pre-processed to enable automatic region-of-interest recognition. Instructions are sent to the actuator based on the lane direction. The authors made the virtually unrealistic assumption that there are no barriers along the road. Yasin et al. [30] detect the obstacle using a cost-efficient and easily available ultrasonic sensor and reduce the speed of the unmanned ground vehicle if the object detected is at a distance greater than 20 ms. If the object is at a distance of less than 20 ms, then the vehicle is stopped or rerouted based on the location and size of the obstacle. The proposed work may not be applicable for critical situations or even for a lane with multiple autonomous vehicles due to the limitations of the sonic sensor. Teli et al. [31] proposed a fuzzy method to avoid local minima in a mobile robot that uses artificial potential field based path planning. Li et al. [32] proposed mixed traffic model taking into account human vehicle drivers behavior and interaction between human vehicle and autonomous vehicles.

Wu et al. [33] presented a human intervention based training process in which humans intervene whenever a autonomous vehicle engages in a disobeying action. The presented method is observed to reduce the convergence time of the DRL algorithm. Baheri [34] aimed at the safety of the autonomous vehicle and incorporated an additional safety module that predicts the trajectory. The safety module is trained offline using historical data and the proposed method is simulated for a three-lane scenario in a high-way.

The feature of DRL to adapt itself to a new scenario makes it a best choice in designing the control system for autonomous vehicles as depicted in the Table 2. Most of the existing literature focuses on one or atmost two of the scenarios like the safety, lane keeping, obstacle detection under various weather or illumination conditions with minimum number of sensors, trajectory prediction, etc. Nevertheless, autonomous vehicle needs to be trained for all scenarios

Table 2 Summary of related works

Work by	Contribution	Tool	Algorithm/Platform used for implementation
Zhao et al., 2020 [19]	Safe driving policy for a highway scenario with 20 vehicles distributed randomly	SUMO	DDQN
Zhang et al., 2018 [20]	Speed regulation by training the vehicle with acquired realistic data	Quadro M400 GPU	DDQN
Chopra et al., 2020 [21]	Automation of driving using raw pixel data along with sensor information	TORCS	DQN
Zhu et al., 2022 [24]	Safe driving policy to follow a leading vehicle by training using new generation simulation data set	Not mentioned	DDPG
Cao et al., 2022 [28]	Safe driving policy for a 3-lane highway	Not mentioned	RL
Kumar et al., 2022 [29]	Vision based safe driving in a obstacle-free track	set up using Pentium core i3 CPU, Rasberry Pi arduino uno	C and Python script

before bringing it on to the road. The proposed work focuses on vehicle control under following scenarios: lane keeping in a two-lane road, driving control under obstacle detection, overtaking of a moving as well as static vehicle, vehicle control based on detected overtaking vehicle and all these are discussed in detail in the next section.

4 Proposed methodology

The primary objective of the proposed research is to drive the autonomous vehicle safely in single as well as multi-agent scenarios with no or reduced collision with the detected static as well as moving objects. The components of DRL models and decision strategies are discussed below.

4.1 Reinforcement learning model

The elements of the proposed RL model which are state space, action space, and reward function as well as their interaction with surrounding environments is explained in the subsequent section.

4.1.1 State space

The sensor inputs decide the present state s_t of the autonomous vehicle. The present state at time t is decided by the current velocity of the agent: v_t in km/h, tangential angle of the agent with respect to road in radians: θ_{dt}, the horizontal distance between the centre of the lane and the agent: δ_t in meters, distance between lane edge and agent: ζ_t in meters, distance between the agent and vehicles or obstacle ahead: $DistF_t$ in meters, and the distance between the agent and obstacles on the right side of the agent: $DistR_t$ in meters, as summarized in Table 3. According to the agent state, the actor-network encodes a state to action at time t. This action is sent to the simulator through shared memory to control the agent and return to the next state, the reward for the same. In each iteration, the set $\langle state, action, reward, next\_state \rangle$ is collected and stored in the replay buffer for network training.

The velocity vector of the agent is divided into three orientations x, y, z ϵR^3. The offset $\delta \epsilon R^1$ is the distance between the agent and the centre of the track which is normalized between $(-1, 1)$The distance between the agent and the edge of the lane $\zeta \epsilon R^{19}$ is measured in $(-45, -19, -12, -7, -4, -2.5, -1.7, -1, -0.5, 0.5, 1, 1.7, 2.5, 4, 7, 12, 19, 45)$ degrees with respect to front of the agent. The parameter $\theta \epsilon R^1$ is the angle between the direction of the track and the direction of the agent and is measured in radians between $(-\pi, \pi)$. Four sensors are spread across the front of the agent and detect obstacles ahead up to a distance of 200 ms. Therefore, $DistF_t$ consists of a set of 4 values. $DistR_t$ consists of a set of 14 values with a maximum range of 200 ms as 14 sensors are located on the right side of the agent and are used to detect obstacles on the right side.

4.1.2 Action space

Agent action space is $a_t = (accelerate, brake, steer)$. Acceleration value of 1 implies that the agent has applied full throttle i.e., in human driving terminology, it has pressed the accelerator pedal completely. If the acceleration value is 0, it implies that the agent has completely let go of the accelerator pedal and it is moving only due to its momentum. A brake value of 0 states that the brake pedal is not at all pressed and a value of 1 implies that it is completely pressed. The steering value of -1 signifies a full right and $+1$ signifies a full left turn.

4.1.3 Reward function

Designing a suitable reward function is one of the key tasks. This function needs to be correctly set so that the agent has a clear understanding of its priorities and learns to drive safely. To briefly explain a scenario, consider an autonomous agent X which is steadily going in its lane and is getting positive rewards. Suppose X detects some other vehicle, named Y, which is seemingly rushing towards X and it might end up crashing onto the vehicle X, then ego-vehicle (X) must receive higher rewards if it decides to take an action of applying brake and maneuvering it away from Y. In order to achieve the objective of lane keep, the agent needs to be trained with an altered reward function wherein it must get a higher reward for

Table 3 Factors that influence the present state

Symbol	Symbol description	Range
$v(km/h)$	Agent velocity	$(0, 300)\epsilon R^3$
δ	Horizontal offset between centre of the lane and the agent	$(-1, 1)\epsilon R^1$
$\theta_d(radian)$	Departure angle between the lane and agent	$(-\pi, \pi)\epsilon R^1$
$\zeta(degree)$	Angle between edge of the lane and agent	$(-45^0, 45^0)\epsilon R^{19}$
$DistF(m)$	Distance between the agent and obstacle(s) ahead of it	$(0, 200)\epsilon R^4$
$DistR(m)$	Distance between the agent and obstacle(s) on the right side	$(0, 200)\epsilon R^{14}$

staying in its lane and a lesser reward for swerving away from the required lane. Another possible approach here is to make use of the sensors to limit the agent to one lane thereby creating a pseudo boundary for the agent. The reward functions are given by Eq. 2, where θ_d is the departure angle, v is the speed of the agent and δ is the offset between the agent and centre of the lane as R_{lane_keep}, $R_{front_collision}$, $R_{overtake}$ and $R_{side_collison}$ are defined by Eqs. 3, 4, 5, and 6 respectively.

$$R = R_{lane\_keep} + R_{front\_collision} + R_{overtake} + R_{side\_collison} \tag{2}$$

$$R_{lane\_keep} = vcos(\theta_d) - vsin(\theta_d) - v|\delta| \tag{3}$$

If the departure angle is less, then $cos(\theta_d)$ will be high while $sin(\theta_d)$ will be a minute value. If the agent is near the centre of the lane, only a small value will be subtracted. Suppose if the vehicle is near the edge of the lane, a higher value will be subtracted from the reward function.

The second part of the work involves overtaking, and collision avoidance with front and right-side obstacles. The reward function for the agent behavior during these scenarios is defined by 4, 5, and 6.

$$R_{overtake} = \begin{cases} 5, & \text{if agent overtakes without collision} \\ 0, & \text{No overtake} \\ -10, & \text{if agent overtakes and collides} \end{cases} \tag{4}$$

$$R_{front\_collision} = \begin{cases} 5, & \textit{if agent avoids collision} \\ 0, & \textit{No obstacle detected} \\ -10, & \textit{if collides with front obstacle} \end{cases} \tag{5}$$

$$R_{side\_collision} = \begin{cases} 5, & \textit{if agent avoids collision} \\ 0, & \textit{No obstacle detected} \\ -10, & \textit{if agent collides with side obstacle} \end{cases} \tag{6}$$

Two different sensors are used for the obstacle detection process, namely *track* and *opponents*. The *track* sensor represented by the ζ value consists of 19 sensors that provide the distance between the centre of the agent and the edge of the lane. These sensor values are stored in a python list in the implementation. The 36 range-finder values given by the *opponents* sensor are also stored in a Python-list. These are located throughout the vehicle at every 10 degrees and they have a maximum range of 200 ms. In the absence of other cars, this sensor outputs the maximum value i.e. 200 ms from each of the sensors. Otherwise, it sends the distance between the agent and other cars present on the track.

4.2 Deep reinforcement learning model

In DRL, both actor and critic networks are represented by neural networks. Neural network is used as a function approximator to map the state to action. The values of the six parameters specified in Table 3 are the inputs to the fully-connected, 4-layered actor-network. The output at the output layer of the actor network specifies the action a_t to be taken, i.e., to brake, accelerate, or steer. Identical network architectures are used for actor and target-actor networks. The critic network, which takes a state-action pair as input, is connected to two fully-connected layers: one to encode the state and the other to encode the action. The critic network's output is the value of the $Q(s_t, a_t)$ function.

4.3 Sequential decision making strategies

Carla, TORCS, Airsim, and SUMO are a few simulators that are considered for autonomous driving applications. Each of them has its own pros and cons. Carla and Airsim, being relatively new software, are more suitable for vision-based driving and are known for their photo-realistic driving scenes, which require higher graphic computations. SUMO does not have its RL capabilities very well developed. On the other hand, TORCS satisfies the requirements to implement this project. Hence, TORCS − 1.3.7 in Ubuntu 16.04 operating system with Python 3.5, Tensorflow, and Keras DL framework is used. TORCS is an open-source, lightweight simulator software that is preferred due to its simplicity and variety of driving dynamics. The server code to control the car is installed with the software during installation. The client code to communicate with the server is written in python language, which offers immense libraries to code RL algorithms. The simulator offers a plethora of sensors and readings. Only the required sensors need to be activated, lest they throw too many unwanted values during the race. Once these sensor readings are interpreted, they need to be fed into the client code to establish control over the car during execution. The proposed RL model is simulated using a TORCS for various scenarios, and each scenario is discussed in detail below.

4.3.1 Lane keep

In case the agent is driving on the left edge of the left lane, *trackPos* sensor outputs a value of +1 and and a value of −1 if it is on the right edge of the left lane. If the car needs to be at the centre of the left lane, it must constantly maintain a value of around 0. Due to tight corners, the presence

of obstacles, or even during overtaking, the agent cannot maintain driving at the centre of the lane at all times. Two methods are implemented to enable lane keeping:

- The first one is a part of the reward function. The third term in the reward function is $v|\delta|$. If the agent is driving at the centre of the lane, δ will be almost equal to 0. In case it has deviated a lot from the centre, $v|\delta|$ a considerably large value will be subtracted from the reward function. When the reward has been reduced, the agent learns that it is not the optimal path to follow and tries to increase it by diligently following the same path in the future. The first method is very efficient in teaching the agent to drive in the centre of its lane.
- However, the second approach is also implemented to leave no room for error. In this method, a pseudo boundary is created between $+0.2$ and -0.2 corresponding to the *trackPos* sensor. If the agent's *trackPos* value is greater than $+0.2$, it means that the agent is steering left from the centre of the lane. Hence the *steer* value is negated to enable the agent to steer back to the centre of the lane. If the value is lesser than -0.2, it signifies that the agent is steering right from the centre of the lane. Just like it was done earlier, the *steer* value is negated so that the agent can steer back to the centre of the lane.

4.3.2 Collision avoidance with front obstacles

Though the simulator gives the freedom of implementing both left-hand drive and right-hand drive, the former driving style was chosen as it is more common around the world. Four sensors out of the 36 *opponents* sensors are used for front obstacle detection. The obstacle can be a vehicle ahead, any object, or a pedestrian. Whenever the car enters a new state, it waits to receive instructions from the actor-network on how to proceed. During this time interval, all the sensor readings are collected and analyzed to see if any obstacles are present ahead of the vehicle. If any obstacle is detected beyond 50 ms ($DistF > 50$), it is not required to apply brakes as there is ample distance between the agent and the obstacle. In case front *opponents* sensors detect a car within a range of 30–50 ms ahead, the agent stops accelerating i.e. directly sets its value to 0. Setting the value directly to zero is analogous to taking the foot off the accelerator pedal in real-world driving scenarios. At this point, the agent keeps moving due to its momentum and gradually slows down due to friction. After a few seconds, if the obstacle ahead is less than 30 ms, it applies the brakes slightly, i.e. it sets the brake value (in the range 0 to 1) to 0.3 as this is the ideal value to gradually slow down the vehicle. If the value is greater than 0.3, the braking action will involve jerky motion, but if it is less than 0.3, it may not slow down at the desired rate. If the obstacle is less than 20 ms away, it applies brakes a tad bit stronger with a value of 0.7. At this value, the vehicle reduces its speed quickly as the obstacle is not very far, and if it does not apply brakes with this value, the agent might end up very close to the obstacle at a considerably higher speed, after which collision is inevitable. In case the obstacle is less than 10 ms away, it applies full brakes, i.e. sets the brake value to 1, so that it comes to a complete halt.

If the obstacle is detected early enough, braking action is taken in a step-by-step manner over a distance of 50 ms and the vehicle smoothly comes to a halt. In case, the obstacle suddenly comes into the agent's path with less distance between them, the agent applies full brakes to avoid a collision. In such a scenario, braking will not be smooth, but that is secondary since it is more important to avoid a collision.

4.3.3 Overtaking

This is interconnected with front obstacle detection. When driving in a lane and any obstacle appears suddenly, the only thing that comes to the mind of any driver is to apply the brake and stop swiftly, and that is exactly what the agent does too. On the other hand, if an obstacle is detected well in advance, the agent can plan to overtake it. When the distance between the agent and the vehicle ahead *DistF* is around 20 ms, an agent takes another set of *opponent* sensor readings to check if any vehicles are currently in the right lane. same set of sensors are used to detect the obstacles in the right lane as well as the obstacles ahead of it. If it is clear of any traffic, vehicle steers to the right lane by choosing the steer right action. After overtaking a vehicle, it checks if it can come back to its previous lane. If the path is again clear, it steers back to its lane and continues driving.

The agent's speed is limited to 90 km per hour. While overtaking, imagine a scenario where the agent has completed half of the overtaking process by going to the right lane. At this point, if the vehicle that was being overtaken, suddenly increases its speed, and if this is equal to or greater than the speed of the agent, the agent decides to abort the overtaking procedure. The agent cannot measure the speed of other vehicles, instead, it knows that the other vehicle's speed has increased when it senses that it is driving parallelly at its maximum speed with the vehicle that it just tried to overtake. In such a scenario, it slows down a little by setting the acceleration value to zero and checks if it can go back to its previous lane. If the road is clear, it steers left and follows the car that it had intended to overtake.

4.4 Collision avoidance with side obstacles

To detect the obstacles at the sides of the agent, a total of 14 sensors are utilized. If all the vehicles that are plying over a stretch of road meticulously follow their lane, then the agent takes no action. If the agent detects a vehicle approaching

Fig. 4 Track selection screen

Fig. 5 Agent driving on the track

from the sides with a continuously decreasing value of *DistR*, the agent lightly steers away to avoid a collision. The agent moves away from the approaching vehicle only if there is space left on the road. In case it was already driving on the edge of the road and if some vehicle approaches from the sides, then it slows down letting the other vehicle pass. If such a situation occurs in the real world, drivers first press the horns of their vehicles to alert the driver who is deviating from their lane. Since the simulator does not have that privilege, it simply slows down, as that is the best the agent can do to avoid a side-ward collision.

The provision of having microphones at layer 1 to detect the horn sound from other vehicles can be a solution to this problem. The pseucode for overtaking and collision avoidance algorithm is depicted in Algorithm 1.

Algorithm 1 Proposed Methodology

```
while agent has not reached the destination do
    read all sensor values
    if any obstacle(s) ahead of the car then
        if the obstacle(s) ahead is far away then
            if it can be overtaken then
                perform overtaking maneuver
            else
                perform slow braking
            end if
        else if the obstacle(s) ahead is very near then
            perform hard braking
        end if
    else if any obstacle(s) on the left/right side then
        set steer value to move away from it
        if the car has reached the edge of the road then
            apply brakes to slow down or stop
        else
            continue
        end if
    else
        set accelerate, brake and steer value to be on the centre of the lane
    end if
end while
```

Fig. 6 Agent goes off-track

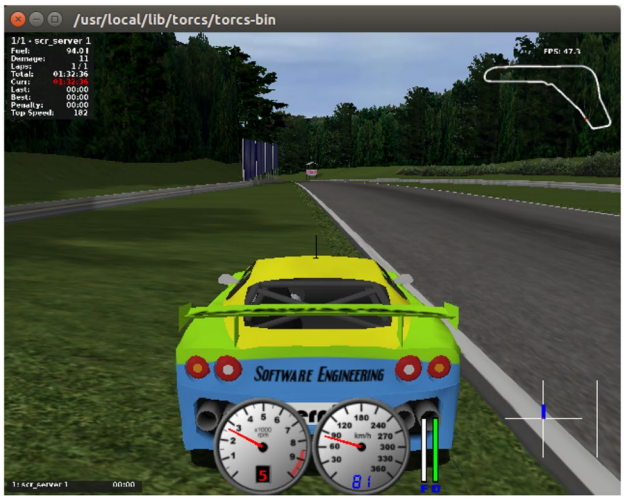

Fig. 7 Agent slowly getting back on track

5 Results

TORCS−1.3.7 is used along with all the python modules required to code RL algorithms. Then, a basic python client code is written to communicate with the TORCS server which enables the user to establish control over the car. Figures 4, 5, 6, and 7 are the screenshots obtained during the execution of the program. Figure 4 shows one of the numerous tracks which is available for selection. Figure 5 portrays a scenario where the agent is driving on the track. Since the agent is not perfectly trained, it sometimes deviates completely off the track, as shown in Fig. 6. However, as soon as it goes off track, its rewards begin to diminish. Hence it soon finds its way back to the track, as shown in Fig. 7.

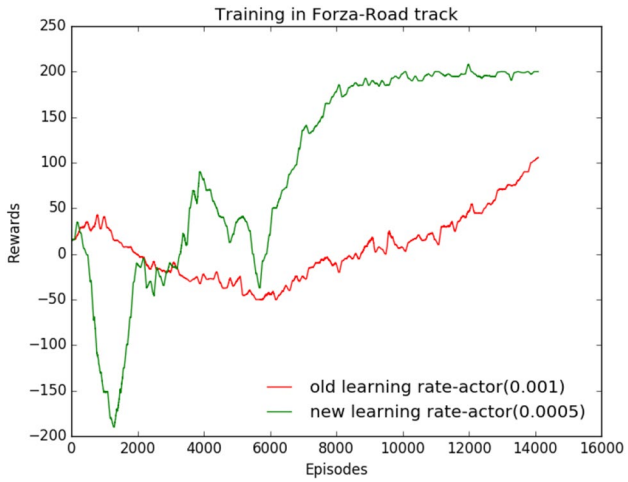

Fig. 8 Training of the actor

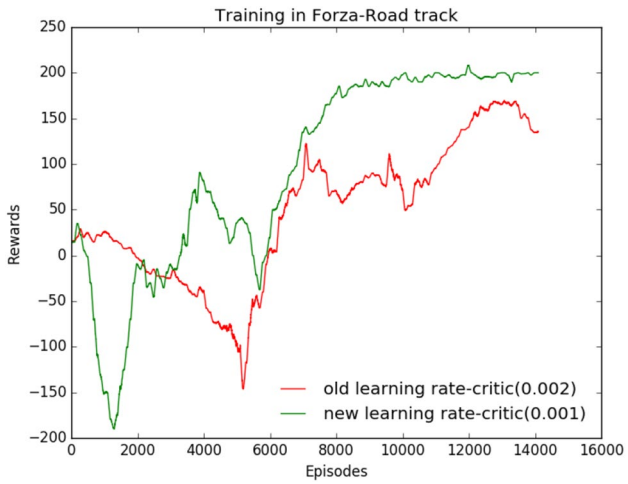

Fig. 9 Training of the critic

Fig. 10 Rewards vs speed, angle, and position (track 1)

After the initial training comprising approximately 8000 iterations, the agent learned to drive on the track but it was not at all moving in a straight line. It was swerving abruptly in its lane. As a solution to this, the learning rate of the actor and critic was altered from 0.001 and 0.002, to 0.0005 and 0.001 respectively to give more time for the agent to learn. The reduced learning rate makes the agent depend more on the past knowledge with less importance to the immediate reward as discussed in Sect. 2. Therefore, the agent takes more time to learn the environment but explores all possible actions. The reduced learning rate resulted in the agent taking around 14,000 iterations to completely re-learn driving from scratch. This is visualized in Fig. 8 and 9. Rewards obtained in each episode of training are plotted against the episode number.

Figure 10 shows the variation of rewards with respect to speed (speedX), distance from lane centre (trackPos) and angle between lane and car (θ_d value). At instances where the speed is less, the rewards follow suit. At times, there are instances where in spite of having sufficiently good speed, the rewards are less. This is due to the fact that the other parameters collectively bring down the rewards as the vehicle may have deviated from the centre of the lane and may have not aligned itself properly with respect to the direction of the lane. In case the rewards are higher than the speed, it shows that the agent has received some extra reward for avoiding some form of probable collision.

Figure 11 further explains the dependency of speed, angle, and distance from the center for rewards. Despite the agent aligning itself properly on the track, it swerved left and right owing to high-speed cornering. On careful observation of Fig. 12, it can be seen that trackPos (distance of the car from the center of the lane) is very large. In this case, it is larger than +1. This easily translates into the fact that the agent was not driving on the road but beside it. This scenario

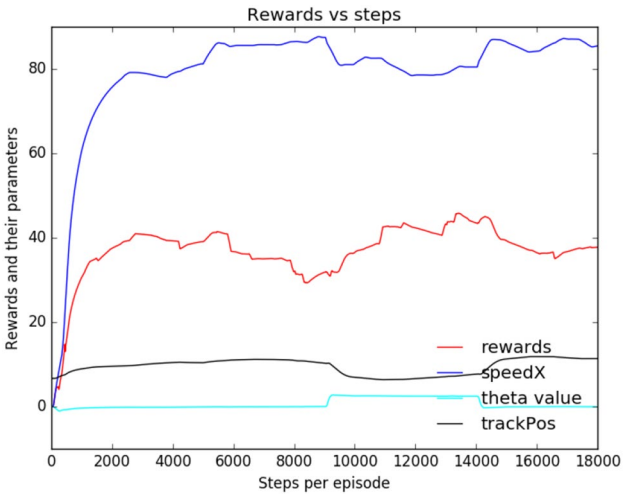

Fig. 12 Rewards vs speed, angle, and position (track 3)

is created specifically for testing how the rewards would look if the agent is made to not drive on the road. Before the agent was trained with an improved learning rate, it was moving in a zig-zag manner in spite of maintaining its lane. Figure 13 shows how the reward varies if the agent is not properly trained to drive itself smoothly.

In all the simulations, the agent successfully managed to drive itself smoothly on the track by constantly interacting with the dynamic environment. It has learned to detect the twists and turns in the track and applies brake accordingly. It efficiently detects obstacles and manages to take precautions or abrupt decisions to avoid collisions. Tesla autonomous cars have improved a lot over the years and have offered a full self driving functionality that drives itself from point A to point B with very little human intervention. but, the car applies an emergency brake if the car

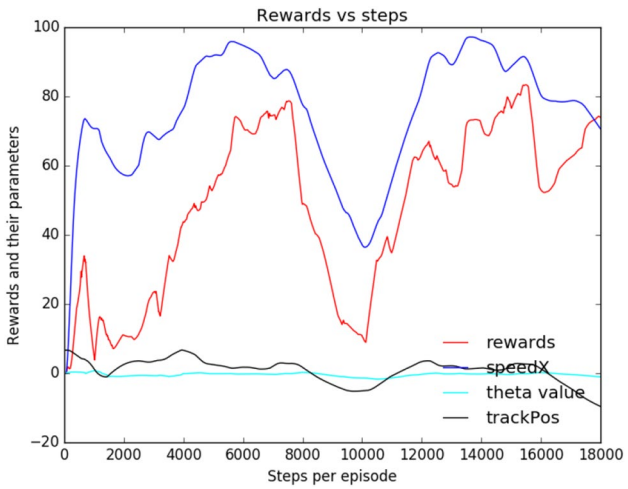

Fig. 11 Rewards vs speed, angle, and position (track 2)

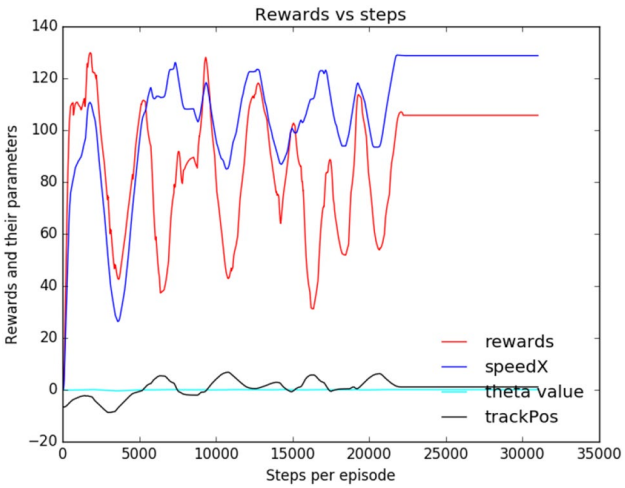

Fig. 13 Rewards vs speed, angle, and position (track 4)

detects any obstacle or have to avoid any collision. After which, the car requires immediate human intervention to proceed [35]. Also until now Tesla cars do not perform any overtaking maneuvers by themselves. If the driver wishes to change lanes over, then they need to first turn on the blinkers. That is when the car begins scanning the side lanes and only if it is free from any traffic, it switches lanes to overtake. In case the driver does not intervene, Teslas just drive in the same lane even though there is slow moving traffic ahead of it. In comparison with this, the proposed work does not require any human intervention at all. When it detects slow moving traffic ahead in the current lane, it automatically begins scanning the side lanes to make a switch. In case, there is some traffic in the other lanes, it waits for them to clear and then switches lane. Unlike in the Tesla, the proposed model constantly keeps scanning the side lanes. Whenever it finds that there is chance to switch and move ahead, it does so giving utmost priority to safety - for the passengers in the car as well as other cars and pedestrians. The proposed work is an improvement even over the work proposed by Huang et. al [36] which focuses only on lane keep. Hence this technology will definitely help in improving autonomous driving and making it more reliable and safe.

6 Conclusion

Over the past few years, there has been a tremendous improvement in technology which is enabling the notion of autonomous vehicles into reality. Elderly people, patients on medication, and persons with disabilities depend on other persons for mobility and can now move around independently. In addition, busy executives can attend to their work undisturbed while traveling and this saves their time. With more and more autonomous vehicles on the road, it can be hoped that traffic accidents can be reduced to a minimum since all these vehicles can be designed to strictly follow traffic rules.

Today, the self-driving vehicle is the need of an hour because of the increasing traffic on the roads. Having self-driven vehicles with the highest possible efficiency, and reliability is required very much for safe traveling. Even though many researchers work on this topic by making use of AI/ML tools, still 100% efficiency has not been achieved. The proposed work uses DDPG for the controlled driving of a vehicle. In the second phase, object detection and collision avoidance are achieved with the help of various sensors present in the vehicle. Thus, the agent can handle situations like front, and side obstacle detection, and overtaking and can maneuver itself to avoid collision.

7 Future scope

The proposed work does not account for variations in the weather, lighting levels, directions given by traffic lights, road conditions, etc. There can be some variations in the sensor values if one or more of these change and it can hamper the performance. It needs to be checked if all these sensors give accurate values after changes in weather and light. If not, it needs to be calibrated accordingly. Since the simulator has no provision to include traffic signals, the same could not be implemented. Therefore, training the agent to recognise the signals and respond appropriately will therefore be a significant effort in the future.

The agent heavily relies on lane markings to detect lanes. In the real world, if the lane markings are erased off in certain areas or if the lane markings are not at all there, it struggles to maintain its lane and drive efficiently. In the future, the agent can be trained to use a camera to detect the edge of the road, and road conditions, and perform seamless smooth driving. The performance of the proposed work can be improved by reducing the training period. The DRL algorithm requires a longer training period to learn all the driving scenarios. The training period can be decreased either by using supervised learning with variety of data along with RL or through human assisted training.

Funding Open access funding provided by Manipal Academy of Higher Education, Manipal.

Data availability We have not used any data set in our work.

Declarations

Conflict of interest The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Kiran BR, Sobh I, Talpaert V, Mannion P, Sallab AAA, Yogamani S, Perez P (2022) Deep reinforcement learning for autonomous driving: A survey. IEEE Transactions on Intelligent Transportation Systems 23(6):4909–4926

2. "The Atlantic," https://www.theatlantic.com/technology/archive/2016/06/beep-beep/489029/, accessed: 2023-01-11
3. "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," On Road Automated-Driving-Committee, Tech. Rep., jun 2018
4. Gupta A, Anpalagan A, Guan L, Khwaja AS (2021) "Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues," *Array*, vol. 10, p. 100057[Online]. Available: https://www.sciencedirect.com/science/article/pii/S2590005621000059
5. Khan MA, Sayed HE, Malik S, Zia T, Khan J, Alkaabi N, Ignatious H (2022) "Level-5 autonomous driving-are we there yet? a review of research literature," *ACM Comput. Surv.*, vol. 55, no. 2, jan. [Online]. Available: https://doi.org/10.1145/3485767
6. Feng D, Haase-Schütz C, Rosenbaum L, Hertlein H, Gläser C, Timm F, Wiesbeck W, Dietmayer K (2021) Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. IEEE Transactions on Intelligent Transportation Systems 22(3):1341–1360
7. Rajagopal BG (2022) Intelligent traffic analysis system for indian road conditions. International Journal of Information Technology 14(4):1733–1745
8. Bojarski M, Testa DD, Dworakowski D, Firner B, Flepp B, Goyal P, Jackel LD, Monfort M, Muller U, Zhang J, Zhang X, Zhao J, Zieba K (2016) "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316
9. Fredriksson T, Mattos DI, Bosch J, Olsson HH (2020) "Data labeling: an empirical investigation into industrial challenges and mitigation strategies," in *International Conference on Product-Focused Software Process Improvement*. Springer, pp. 202–216
10. Sutton RS, Barto AG (1998) Introduction to Reinforcement Learning, 1st edn. MIT Press, Cambridge, MA, USA
11. Naveen Raj R, Nayak A, Kumar MS (2020) "A survey and performance evaluation of reinforcement learning based spectrum aware routing in cognitive radio ad hoc networks," *International Journal of Wireless Information Networks*, vol. 27, no. 1, pp. 144–163
12. Rahmati M, Nadeem M, Sadhu V, Pompili D (2019) "Uw-marl: Multi-agent reinforcement learning for underwater adaptive sampling using autonomous vehicles," in *Proceedings of the International Conference on Underwater Networks & Systems*, pp. 1–5
13. Arulkumaran K, Deisenroth MP, Brundage M, Bharath AA (2017) Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine 34(6):26–38
14. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D (2015) Human-level control through deep reinforcement learning. Nature 518:529–533
15. Rani G, Pandey U, Wagde AA, Dhaka VS (2022) "A deep reinforcement learning technique for bug detection in video games," *International Journal of Information Technology*
16. Coronato A, Naeem M, De Pietro G, Paragliola G (2020) Reinforcement learning for intelligent healthcare applications: A survey. Artificial Intelligence in Medicine 109:101964
17. Luong NC, Hoang DT, Gong S, Niyato D, Wang P, Liang Y-C, Kim DI (2019) Applications of deep reinforcement learning in communications and networking: A survey. IEEE Communications Surveys and Tutorials 21(4):3133–3174
18. Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D (2015) "Continuous control with deep reinforcement learning," *arXiv preprint*arXiv:1509.02971
19. Zhao J, Qu T, Xu F (2020) A deep reinforcement learning approach for autonomous highway driving. IFAC-PapersOnLine 53(5):542–546
20. Zhang Y, Sun P, Yin Y, Lin L, Wang X (2018)"Human-like autonomous vehicle speed control by deep reinforcement learning with double q-learning," in, IEEE Intelligent Vehicles Symposium (IV). IEEE 2018:1251–1256
21. Chopra R, Roy SS (2020) "End-to-end reinforcement learning for self-driving car," in *Advanced computing and intelligent engineering*. Springer, pp. 53–61
22. Elallid BB, Benamar N, Hafid AS, Rachidi T, Mrani N (2022) "A comprehensive survey on the application of deep and reinforcement learning approaches in autonomous driving," *Journal of King Saud University-Computer and Information Sciences*
23. Hussain R, Zeadally S (2018) Autonomous cars: Research results, issues, and future challenges. IEEE Communications Surveys and Tutorials 21(2):1275–1313
24. Zhu M, Wang Y, Pu Z, Hu J, Wang X, Ke R (2020) Safe, efficient, and comfortable velocity control based on reinforcement learning for autonomous driving. Transportation Research Part C: Emerging Technologies 117:102662
25. Omeiza D, Webb H, Jirotka M, Kunze L (2021) "Explanations in autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*
26. Grigorescu S, Trasnea B, Cocias T, Macesanu G (2020) A survey of deep learning techniques for autonomous driving. Journal of Field Robotics 37(3):362–386
27. Ma Y, Wang Z, Yang H, Yang L (2020) Artificial intelligence applications in the development of autonomous vehicles: a survey. IEEE/CAA Journal of Automatica Sinica 7(2):315–329
28. Cao Z, Xu S, Jiao X, Peng H, Yang D (2022) Trustworthy safety improvement for autonomous driving using reinforcement learning. Transportation research part C: emerging technologies 138:103656
29. Kumar A, Saini T, Pandey PB, Agarwal A, Agrawal A, Agarwal B (2022) Vision-based outdoor navigation of self-driving car using lane detection. International Journal of Information Technology 14(1):215–227
30. Yasin JN, Mohammed SAS, Haghayan M, Heikkown J, Tenhumen H, Plosila J (2022) Low-cost ultrasonic based object detection and collision avoidance method for autonomous robots. International Journal of Information Technology 13(1):97–107
31. Teli TA, Wani MA (2021) A fuzzy based local minima avoidance path planning in autonomous robots. International Journal of Information Technology 13(1):33–40
32. Li X, Xiao Y, Zhao X, Ma X, Wang X (2023) Modeling mixed traffic flows of human-driving vehicles and connected and autonomous vehicles considering human drivers' cognitive characteristics and driving behavior interaction. Physica A: Statistical Mechanics and its Applications 609:128368
33. Wu J, Huang Z, Hu Z, Lv C (2022) "Toward human-in-the-loop AI: Enhancing deep reinforcement learning via real-time human guidance for autonomous driving," *Engineering*
34. Baheri A (2022) Safe reinforcement learning with mixture density network, with application to autonomous driving. Results in Control and Optimization 6:100095
35. Dikmen M, Burns C (2017) "Trust in autonomous vehicles: The case of tesla autopilot and summon," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1093–1098
36. Huang Z, Zhang J, Tian R, Zhang Y (2019) "End-to-end autonomous driving decision based on deep reinforcement learning," in *2019 5th International Conference on Control, Automation and Robotics (ICCAR)*. IEEE, pp. 658–662