IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 22 August 2023, accepted 28 October 2023, date of publication 6 November 2023, date of current version 10 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3330665

RESEARCH ARTICLE

An Empirical Study of DDPG and PPO-Based Reinforcement Learning Algorithms for Autonomous Driving

SANJNA SIBOO[1], ANUSHKA BHATTACHARYYA[1], RASHMI NAVEEN RAJ[1], (Member, IEEE), AND S. H. ASHWIN[2]

[1]Department of Information and Communication Technology, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, Karnataka 576104, India
[2]Standard Chartered Global Business Services, Bengaluru 560103, India

Corresponding author: Rashmi Naveen Raj (rashmi.naveen@manipal.edu)

This work was supported by the Manipal Academy of Higher Education, Manipal.

ABSTRACT Autonomous vehicles mitigate road accidents and provide safe transportation with a smooth traffic flow. They are expected to greatly improve the quality of the elderly or people with impairments by improving their mobility due to the ease of access to transportation. Autonomous vehicles sense the driving environment and navigate through it without human intervention. And, Deep Reinforcement Learning (DRL) is witnessed as a powerful machine learning solution to address a sequential decision problem in autonomous vehicles. The detailed state-of-the-art work in autonomous vehicles using DRL algorithms along with future research directions is discussed. Due to the high dimensional action space, two continuous action space DRL algorithms: Deterministic Policy Gradient (DDPG) and Proximal Policy Optimization (PPO) are chosen to address the complex autonomous driving problem. The proposed DDPG and PPO based decision-making models are trained and tested using the TORC simulator. Both the algorithms are trained for the same number of episodes for lane keeping as well as multi-agent collision avoidance scenarios. To the best of our knowledge, this is the first paper to present the comparative performance analysis of these two algorithms, and DDPG is found to perform better in terms of higher reward and faster convergence than PPO. Hence, DDPG is a suitable option in the design of a decision model for autonomous driving.

INDEX TERMS Machine learning, deep reinforcement learning, deep deterministic policy gradient, continuous space, autonomous vehicles, proximal policy optimization.

I. INTRODUCTION

Autonomous vehicles also known as driverless vehicles navigate through the environment without any human intervention. These vehicles use different sensors such as LiDAR, cameras, GNSS, sonar, etc. to understand various environmental conditions and take actions accordingly [1]. Different control systems for brakes, lane keeping, steering, etc. work together using sensory information to decide on different possible trajectories and strategies. The expeditious technological development in sensor technology, image processing, control system design, and inter-vehicular communication systems has enabled progress in reliable autonomous driving.

Generally, road vehicles are classified into 6 levels from level 0 to 5 based on their autonomous capabilities by SAE International [2]. The first three levels (0 to 2) depend more on human driving with almost no automated assistance to the driver. For example, Tesla cars provide level 2 automated assistance. The next three levels (3 to 5) are automated drivers with little to no human assistance. Some major developments in the field include Waymo providing self-driving taxi services (with SAE level 4) with fully driverless taxis in 2020. Cruise, under General Motors, has developed a level 4 and 5 EV named Origin. Another company, Argon AI is working with Volkswagen and Ford in the development of level 4 software [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Wang.

The need for autonomous driving arises from several different factors including safety, reduced road accidents due to human error, improved traffic flow, efficiency, and convenience. Human error is a key cause of many car accidents, and autonomous vehicles can reduce this risk by relying on sensors and complex algorithms to make decisions and adapt to road conditions [4]. Another advantage is it increases mobility for those who cannot drive themselves, such as the elderly or people with impairments. Self-driving cars can greatly improve the quality of their lives by giving them access to transportation. Last but not least, autonomous driving technology can aid in easing traffic congestion and enhancing overall road efficiency. The standard building blocks of autonomous driving systems are:

Sensing: A modern autonomous driving system's sensor architecture has been notable for including multiple sets of cameras and LiDARs etc for providing the 3D position of the vehicle. The sonar sensors that can detect the siren of public safety vehicles should also be included as input sensors.

Perception and Localization: The perception module's goal is to create an intermediate-level depiction of the environmental state that will later be used by the system to eventually create the driving policy.

Scene Representation: The main aim of sensor fusion is to generalize to situations where the environment becomes more abstract. Mapping is a critical component of automated driving. Maps also play a vital role as vehicles' current location can be pinpointed within the map. Due to the magnitude of the problem, traditional mapping techniques are supplemented with semantic object detection to provide reliable disambiguation.

Planning and Decision Making: The trajectory planning module is a critical component of the autonomous driving pipeline. This module must generate motion-level commands that steer the agent given a route-level plan-generated map.

Control: A controller defines various actions required at each point along the path based on a predetermined map or expert driving recording of the same values at each way-point.

Early autonomous vehicle systems relied significantly on precise sensory data to provide accurate environment perception. The control of these self-driving vehicles was handled by rule-based controllers, the parameters of which were set by the developers and manually adjusted after simulation and field testing. The disadvantage of this approach was that it was time-consuming and difficult to generalize to new scenarios [4]. Autonomous vehicles should have a control system to manage different traffic conditions such as highways, traffic jams, the presence of emergency vehicles, and public safety systems like ambulances, fire safety, or police vehicles, etc.

Machine Learning (ML) plays a crucial role in the development and operation of autonomous vehicles by enabling them to perceive, interpret, and make decisions based on complex data they receive from their environment. ML algorithms are usually classified into three broad categories: supervised, unsupervised, and Reinforcement Learning (RL). In supervised learning, the model is trained using labeled data available beforehand. The algorithm then learns from this labeled data to generalize patterns and relationships, enabling it to make accurate predictions on new, unseen data. Opposite to that of supervised, unsupervised deals with learning unlabeled data. The third category is RL, where an agent improves its performance in a task by constantly interacting with its surroundings.

RL was introduced to incorporate ambient intelligence into autonomous systems to address the aforementioned issue. RL can be used to solve many real-world problems [5], [6] but sometimes, there exist certain situations where conventional RL algorithms fail to provide desirable results. This is mainly because the state and/or action spaces involved in these problems are very high-dimensional. Researchers have investigated various RL algorithms in the field of autonomous vehicles [7], [8], [9], [10], [11]. When it comes to traditional RL algorithms, it is not feasible to store and map all the state-action pairs. Hence, there is a need to use neural networks and inculcate Deep Learning (DL) into RL. The neural networks help estimate the states and the possible state-action pairs. This makes the solution more manageable and generalizes an otherwise immeasurable state space. To make it more efficient in dealing with high dimensional sensory data, Deep Reinforcement Learning (DRL) with continuous state and action spaces has been developed. The usual approach to discretize continuous variables is often not feasible since it leads to dimensionality issues termed as 'Curse of Dimensionality'. These issues arise while dealing with high dimensional data and action/state spaces, requiring more time and expensive resource utilization. The exceptional performance of DL techniques has expanded its application in autonomous vehicles [4]. To the best of our knowledge, this is the first paper to compare the performance of Deep Deterministic Policy Gradient (DDPG) and recently developed Proximal Policy Optimisation (PPO) based DRL algorithms. The contributions of the research are:

- Various DRL-based research papers are discussed and contributions are highlighted.
- DDPG and PPO oriented state-of-the-art developments are discussed
- A comparative performance analysis of DDPG and PPO algorithms observed through The Open Racing Car Simulator (TORCS) is presented.
- Future scope in autonomous vehicles is also discussed.

The rest of the paper is organized as follows: section II covers the background and functioning of the DRL algorithms. In section III, a comprehensive review of research related to this topic is presented and summarised in a table. Section IV defines the elements of the RL model and discusses the performance analysis of DDPG and PPO based algorithms in lane keeping, obstacle detection, and overtaking scenarios. Section V covers future scope with section VI concludes this paper.

II. PRELIMINARIES

A general overview of the basics of RL and DL along with its different elements and various DRL algorithms is provided in this section.

RL consists of three main elements: the agent, the environment, and the reward. The agent is driven by the algorithm to interact with the environment, and in turn, gain feedback from the environment in the form of a reward. An RL problem can be expressed as a Markov Decision Process (MDP), which consists of the following factors:

S : Set of all states occurring in the environment, $s \in S$.

A: Set of all actions that can possibly be taken by the agent, $a \in A$ to perform.

$p(s_{t+1}|s_t, a_t)$: State transition model describing the change in environment state depending on the action taken a_t by an agent in the current state s_t.

$p(r_{t+1}|s_t, a_t)$: Reward model that describes the reward value r_{t+1} that the agent receives from the environment as feedback for taking a certain action in that time step.

γ: The discount factor that controls the importance of future rewards.

DL is a type of ML with multiple layers of neural networks. The term deep is due to the presence of multiple layers. DL has the potential to automatically extract the feature sets from the data set and perform tasks like object detection, image classification, etc., without the need for many manual sequential steps like data pre-processing, feature selection, classification, etc. as required in ML [12], [13].

The RL algorithm aims to maximize the reward (based on the context of the problem) by driving the agent from one state to another through actions and adjusting the actions depending on the rewards received from the environment in turn. The use of DL with RL has extended the scope of RL to a variety of challenging applications. The three commonly used DRL algorithms for autonomous driving are Deep Q Network (DQN), DDPG, and PPO. DQN deals with discrete state-action spaces and is more suitable for low-dimensional action space. DDPG and PPO are continuous space DRL algorithms that are suitable for most real-world problems.

A. DEEP Q NETWORK

DeepMind first proposed DQN or Deep-Q Networks in 2015 in an effort to bring the benefits of DL to RL [14]. When the state space is discrete, Q-networks can quickly formulate it into a table. However, as the number of states increases, or when the states are continuous, it becomes impossible to search all possible state-action pairs to find the optimal value of Q. This is where DQN comes into play, which uses deep neural networks to parameterize the Q-function as Q(s, a, w). The state is given as input, and all possible actions' Q-value is generated as output. The state space can be extended to higher dimensions or even made continuous using neural networks.

B. DEEP DETERMINISTIC POLICY GRADIENT ALGORITHM

DDPG is a model-free, off-policy RL algorithm specifically designed for problems with continuous action spaces. It is an extension of the well-known Deterministic Policy Gradient (DPG) algorithm and uses deep neural networks to represent the policy and value function.

When the execution starts the current state s of the agent is fed into the actor network. The actor-network selects the desired action denoted by a and sends it back to the agent. The agent performs that particular action, obtains a reward r (using the reward function), and then moves on to the next state. The entire experience consisting of (s, a, r, s') is stored in an experience relay buffer. Random sampling is done on the buffer, after which a few samples are stored in a mini-batch. The reason why random sampling is done is to ensure that the agent not only focuses on recent experiences but also on samples from a diverse set of experiences. This helps to prevent over-fitting, where the agent only learns to respond to specific situations and is unable to generalize to new situations.

From the mini-batch, the current state value s, is fed into the critic network. The main goal of the critic network is to evaluate how efficient the action was in that particular state. The critic network calculates the Q-value for the state-action pair denoted by $Q(s, a)$ using Equation 1.

$$Q^*(s, a) = E[r(s, a) + \gamma \max Q(s', a')] \tag{1}$$

where $r(s, a)$ is the reward obtained by taking action a in state s, γ is the discount factor, which determines the importance of future rewards, $\mu(s')$ is the policy of the actor-network, which specifies the best action to take in-state s' according to the current estimate of the Q-value function.

The next state value s' from the mini-batch is then fed into the target actor network. The main goal of the target actor network is to output an action a' that is best suited in state s'.

The next state and action value is then passed onto the target critic network which evaluates the action generated by the target actor and calculates $Q(s', a')$.

Using values $Q(s, a)$ and $Q(s', a')$ the loss function is calculated as shown in Equation 2 which is used to update the critic network and helps it evaluate the actions in a better way.

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

Additionally, the policy gradient is calculated using the Q value of (s, a), which is then used to update the actor network. This enables the actor to make improved decisions.

The working of the algorithm in the TORCS environment is shown in Figure 1.

C. PROXIMAL POLICY OPTIMIZATION

PPO was introduced in 2017 [15], and aims to give the reliability of Trust Region Policy Optimisation (TRPO) while using only first-order optimization. It is based on the policy gradient methods, which work on policy and there is no replay buffer, and a batch of experiences that are used to update the policy are discarded. The objective function of policy

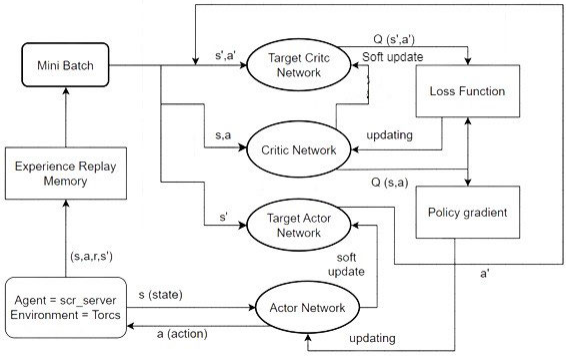


FIGURE 1. DDPG block diagram.

gradient methods is shown in Equation 3

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta \left(a_t \mid s_t \right) \hat{A}_t \right] \tag{3}$$

the first term is the log of the output of the policy, which is probabilities of actions based on observed states(which are taken as the input). The second term $\hat{A}_t$ denotes the advantage of taking a specific action. It's computed as the difference between the actual reward of taking action and the expected reward. Hence if it is positive, then the probability of the action taken gets increased according to the Equation 3 and vice versa. This intuitively means that the policy, rather than giving a 100% probability to an action, slowly increases or decreases its probability of happening according to the rewards and advantages.

To prevent the updates from changing the policy too abruptly and producing unexpected trajectories, the concept of TRPO was introduced. The objective now is defined as shown in Equation 4

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta \left(a_t \mid s_t \right)}{\pi_{\theta_{\text{old}}} \left(a_t \mid s_t \right)} \hat{A}_t \right] = \hat{\mathbb{E}}_t \left[r_t(\theta) \hat{A}_t \right] \tag{4}$$

Here, instead of a log, the ratio of the updated policy outputs and the old policy outputs is considered. This is denoted by $r_t(\theta)$. This ratio would be greater than 1 when the action a_t is more likely with the newer policy and between 0 and 1 if the action is less likely.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip} \left(r_t(\theta), 1 - \epsilon, 1 + \epsilon \right) \hat{A}_t \right) \right] \tag{5}$$

Basically, this clipping function is used to limit the effect of the gradient update so that the probability of an action doesn't become too high or too low, leading to unpredictable behaviors in the trajectories. This clipping ensures that, with each update, the new policy doesn't stray too far from the previous policy and hence prevents the policy from going downhill at a certain step.

The actor uses the old and new policy to get the ratio of the probability of a certain action given each policy. The critic gives the value function for calculating the advantage. Both ratio and advantage are then used in the clipped objective function to give a new policy update to the actor. The critic also gives a value function for TD error which is then used

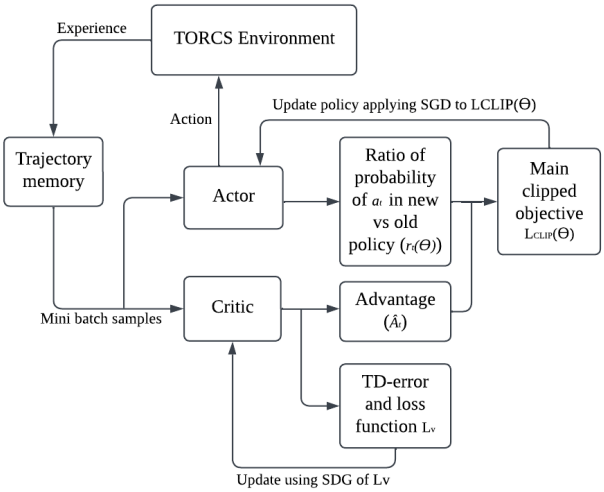


FIGURE 2. PPO block diagram.

in loss function L_v which updates the new value functions for the critic. The block diagram of PPO is shown in Figure 2

III. LITERATURE SURVEY

A lot of advancements and research are going on in the field of autonomous vehicles. A comprehensive literature review has been done with three different sections. Section III-A covers research in different aspects of autonomous driving, algorithms, and methods generally used. Sections III-B and III-C focus more on articles that have used DDPG and PPO algorithms for lane-keeping implementations.

A. GENERAL BASED PAPERS

Chae et al. [8] proposed a braking system that utilized DRL. The system determines when brakes are to be applied in real world by assessing the collision risk based on sensor data. For the brake control policy, the authors employ a DQN and conduct computer simulations. The experimental results demonstrate that the control agent exhibits desirable actions, successfully avoiding collisions without any mistakes in diverse unknown territory. Safety tests were conducted and it was observed that the model could successfully avoid collision for time to collision values above 1.5s. Ashwin and Naveen Raj [16] proposed a DQN based algorithm for various overtaking scenarios with collision avoidance.

Antonio and Maria-Dolores [17] attempted to improve safety by predicting the 5G communication network latency on the intersection management system of autonomous vehicles. Xia et al. [9] proposed a new control strategy that combines the experience of professional drivers with a Q-learning algorithm that uses filtered experience replay. The experimental results demonstrated on TORCS showed that their proposed model significantly reduced the time required for learning by 71.2% compared to the existing neural-fitted Q-iteration algorithm. Additionally, the stability of self-driving vehicles increases by approximately 32% in comparison.

The work by Zhang et al. [18] focused on using Double DQN (DDQN) to build the vehicle speed control model.

In Q-Learning, an agent selects an action with the best Q-value, which might not be an optimal choice. To resolve the overestimation, two separate Q-value estimators (Double Q) are used. The authors also reported that the DDQN model's score was very high compared to that of the DQN algorithm. Li et al. [19] designed a DRL approach to prevent collisions in different intersection scenarios while ensuring optimal travel efficiency. The paper used the DQN algorithm to train their DRL model. Three crossing scenarios with the most accident-prone intersections were designed on CARLA to test this model. The proposed method gave a more consistent and stable result than the other two algorithms, i.e. Monte-Carlo-sampling-based method and the Bayesian-network-based method, which showed an overly conservative approach.

Karthikeyan et al. [20] introduced a novel method to navigate intersections called DRL-based autonomous intersection management. This model consists of two components: the brake-safe control model and the intersection controller. The intersection controller comprises the environment, agent, priority assignment model, and queue. The priority assignment model uses the DQN algorithm to decide which vehicles should be allowed to pass or forbidden. The authors tried various different reward functions and found one of those functions to have an 83 percent improvement in throughput for navigating intersections. Sallab et al. [21] proposed different methods for an end-to-end autonomous driving model (lane-keeping assistance) that took raw sensor inputs and outputs driving actions. Two algorithms Deep Deterministic Actor-Critic Method (DDAC) and DQN were used for this purpose. The algorithms were tested on tracks that contained both straight and curved parts. For straight tracks, the performance of both the algorithms was the same but for the curved part of the track performance of DDAC was much better than DQN. Thus DDAC algorithm has an excellent performance on both straight and curved parts of the track.

He et al. [22] proposed an improved version of the DDPG algorithm using double critic networks and a priority experience replay mechanism. This enhanced algorithm was then applied to develop a lane-following method. Experimental results conducted on the TORCS platform demonstrated that the proposed algorithm achieved excellent lane-following results under various road conditions. Wang et al. [23] proposed two strategies using different DRL algorithms for lane-keeping assistance. The first algorithm used was the DQN, which operates with a discrete action space. The second algorithm employed was DDPG, which operates with a continuous action space. The environment as well as the agent dynamics are also modeled in simulink. Comparing the two algorithms, it was observed that the DDPG agent performed better, as it ran fewer episodes, took less time, and achieved a higher average reward.

Sharma et al. [11] proposed a DL technique for implementing both lateral and longitudinal control in autonomous vehicles. TORCS simulator was used for evaluating the implementation. Based on the road course, two different neural networks were trained to predict vehicle speed and steering. The trained model could complete an entire lap without going off track. The limitation in this paper was the fact that the speed and steering networks were not able to obtain 100 % efficiency in other tracks. More data from different tracks was required in order to make the technique more generalized to different tracks. Radwan et al. [24] proposed a paper to obtain the most optimal algorithm for the autonomous vehicle to avoid hurdles or barriers in a 3D environment obtained through the Unity game engine. The training was done using two algorithms: Soft Actor-Critic (SAC) and PPO. While training with and without obstacles, PPO showed better performance and obtained greater rewards than SAC. Future work included the addition of recurrent neural networks to extract features from camera sensors.

Li et al. [10] proposed a framework for autonomous vehicles to make decisions in lane-changing scenarios using DRL. This method incorporated risk awareness into the reward function of any DRL algorithm for improving the safety of the vehicle. The framework was trained on CARLA simulator under two circumstances (one with stationary obstacles and the other with mobile obstacles). The proposed method outperformed other traditional methods in terms of performance, safety, and efficiency. Elallid et al. [25] proposed an RL-based technique using DQN to control autonomous vehicles in challenging scenarios involving pedestrians and moving vehicles. The model was tested on CARLA simulator. The reward function was designed in such a way that the model returned -200 if there was any collision with the vehicle/pedestrian and 100 if the agent reached the required destination. The results showed that the collision rate after 5000 episodes had decreased and the vehicle was able to avoid crashing with pedestrians and moving vehicles.

Sallab et al. [26] proposed a framework that incorporated Recurrent Neural Networks along with DRL algorithms (DQN and DDAC) enabling the vehicle to handle scenarios with imperfect observations. The model was tested in an open-source car racing simulator ie TORCS for lane-keeping scenarios. Both the algorithms showed successful results, however, DDAC delivered better performance due to the presence of continuous policy which helped choose smoother actions. Alzubaidi et al. [27] proposed an Emergency Vehicle (EMVs) aware Lane Changing Decision (LCD) model using DQN. This algorithm was used for training and testing since the addressed issue had infinite state spaces. The framework was tested on an open-source implementation known as Highway-en which could be used to easily create custom actions and observations for autonomous vehicles. The results proved that the proposed technique was more efficient than the rule-based Minimizing Overall Braking Induced by Lane (MOBIL) since the agent in the former case took lesser time to change lanes when an EMV was approaching and avoided blocking its path.

Fayjie et al. [28] proposed a DRL-based strategy to facilitate navigation and avoid obstacles in an urban environment. DQN was used as the DRL algorithm along with two types of sensors: cameras and lasers in front of the car. The goal of the paper was to maximize the positive rewards obtained during the entire training process. The environment is designed in Utility Game Engine as it allowed training the DQN algorithm to drive the vehicle and take almost real-time driving decisions. Results showed that the developed model was capable of taking correct decisions and avoiding obstacles. Ye et al. [29] proposed a DRL method for decision-making in autonomous vehicles on highways, where the action horizon is continuous. The proposed method incorporates a deep neural network to estimate the value function of the RL algorithm, which allows for continuous action selection. The algorithm was trained using a combination of on-policy and off-policy methods to improve the stability and convergence rate of the learning process. The method was evaluated using a simulated highway scenario and its performance was compared to other baseline methods. The results showed that the proposed approach performed better than the baseline algorithms in terms of safety, efficiency, and comfort.

Huang et al. [30] proposed an efficient deep RL approach for autonomous driving that incorporates imitative expert priors. In RL, the sample complexity can be hugely reduced and learning speed accelerated with the help of human demonstrations. The proposed method uses a Deep Neural Networks (DNN) to approximate the value function of the RL algorithm, which allows for continuous action selection. The algorithm is trained using a combination of expert demonstrations and on-policy methods to improve the stability and convergence rate of the learning process. The proposed method outperformed the baseline algorithms in terms of efficiency and stability.

Baheri et al. [31] presented a safe DRL system where both rule-based and learning-based safety approaches are leveraged. The system consists of two modules, one based on common driving practices and other based on data-driven safety patterns. The data-driven module incorporates a look-ahead beyond immediate rewards to predict safety in future. The DDQN algorithm was combined with this safety system and used to train the RNN used for the agent. This method led to fewer collision and an adaptive decision making system and accelerated the learning process.

Guan et al. [32] presented a centralized coordination scheme of autonomous vehicles at traffic signal-less intersections. Model accelerated PPO (incorporation of a prior model into the algorithm) was proposed to increase the speed of learning. The environmental model is used to generate virtual samples that the algorithm can train on. This leads to a greater sample efficiency for an on-policy algorithm. A set of rules were set for the environment consisting of a four way intersection with single lanes regarding the expected movement of vehicles. This method achieved a human-like policy with optimal traffic efficiency. The model was compared against a coordination scheme based on the model predictive control method. The results showed shorter computation time periods and better efficiency of vehicle management at the intersection.

Holen et al. [33] presented a road detection method that combines road detection algorithms with reinforcement learning. Here, the agent is trained using PPO in a simulated environment. MobileNet, a convolutional neural network used in the road detection algorithm, provides feedback to the DRL algorithm whenever the vehicle is off the road. It was trained and tested on two simulated environments, a Unity environment and a simpler gym environment. Results showed that the agent was able to drive past more checkpoints with the help MobileNet hence improving the training results. Hence combining a road detection algorithm with DRL led to improved results and greater efficiency in training without modifying the environments.

Li et al. [34] proposed a framework that separates the perception and control modules of a vision-based lateral control system. The control module, which is based on reinforcement learning, employs the track attributes predicted by the perception module, which uses a multi-task learning (MTL) neural network to predict them from a driver-view image. To increase data economy, the authors also suggested a DRL environment based on TORCS called Visual TORCS (VTORCS). With the driver-view picture as its input, the suggested framework combines both modules to produce the MTL-RL controller, which can direct the vehicle's run along the track's centre. Cai et al. [35] proposed a deep imitative reinforcement learning technique(DIRL) that combines imitative learning and model based DRL. The method acquires driving knowledge through both learning from human teachers and improving through interaction with a safe offline world model.The key contributions include the introduction of Reveries-net, an uncertainty-aware deep network for learning a probabilistic world model, and the use of DIRL for training an end-to-end policy for visual control. The proposed method outperformed previous IL and RL methods with regard to task performance and sample efficiency.

Chen et al. [36] proposed an interpretable DRL method for end-to-end urban driving. The model takes camera images as well as LiDAR images as input. The method uses a sequential latent environment model to compress high-dimensional observations into a low-dimensional latent space, enabling a semantic birdeye mask to be generated and providing an explanation of how the learned policy behaves. The authors compared it to baselines of DDPG, DQN, SAC and TD3 and observed that it generally outperformed all of these algorithms. Mushtaq et al. [37] proposed a strategy consisting of two stages to regulate traffic in self-driving settings through the utilization of DRL and Smart Re-routing(SR) methods. The DRL implementation is based on DQN. The initial stage focuses on enhancing traffic flow at intersections

by developing a flexible traffic signal control system, whereas the subsequent stage employs an intelligent re-routing technique to divert traffic that approaches intersections towards alternative routes, thus preventing traffic congestion at these locations. Fernando et al. [38] reviewed the current approaches and methods in the field of inverse reinforcement learning(IRL) with regard to the challenges of autonomous driving. Supervised learning models such as CS-LSTM and MATF-GAN, GAIL, L-IRL, and D-IRL were evaluated and discussed. Future possible breakthroughs in the field were also discussed.

Aoki et al. [39] developed a system for autonomous vehicles called a cooperative perception scheme that utilizes DRL to improve the accuracy of detecting surrounding objects. The system optimizes the network load by intelligently selecting the data to transmit. The DRL approach was based on DQN. A special simulation was created using SUMO traffic simulator, CARLA vehicle simulator, and a SUMO-CARLA bridge to convert the mobility data of SUMO for CARLA. Hoel et al. [40] discussed a framework for tactical decision making combining learning and planning. Based on AlphaGo algorithm the method is applied to two highway driving in a simulation. The MOBIL change strategy was utilized for modeling the lane changes made by surrounding vehicles. The framework is shown to outperform commonly used baseline methods, and a comparison with the Monte Carlo tree search and the neural network policy separately proved the advantage of combining planning and learning. The MCTS/NN agent obtained from the proposed framework is shown to be capable and is able to focus the tree search on the most promising regions.

B. DDPG BASED PAPERS

DDPG is one of the significant algorithms in the field of autonomous vehicles. Huang et al. [41] proposed different methods for an end-to-end autonomous driving model (Lane Keeping Assistance) that takes raw sensor inputs and outputs driving actions in a single agent scenario. The proposed work established the exact mapping using the DDPG algorithm on TORCS simulation software.

Gongsheng et al. [42] proposed a vehicle control algorithm designed on a TORCS simulator combined with DDPG. Combined with an actor-critic algorithm, experience playback, and a separate target network, DDPG has more robust effects. This method made driving the car more steady and demonstrated excellent learning in less training time.

This study focused on addressing the challenges of autonomous driving, particularly in scenarios involving lane changes, overtaking, and yielding. Hongsuk Yi [43] proposed a solution using a multi-agent-based deep deterministic policy gradient algorithm to control the actions of autonomous vehicles. When the car reached its target, a positive high reward value was defined, and a negative reward value was defined when it reached the incorrect location or when an accident had occurred. The simulation results demonstrated that the thoroughly trained autonomous vehicles were capable of navigating a continuous lane-changing road environment successfully.

Wang et al. [44] proposed a technique to overcome the limitations of lane-changing behavior with DDPG to ensure safety and comfort. The reward function, which plays a significant role in learning the optimal policy. Despite initially frequently running off the road boundary, the RL vehicle agent ultimately achieves smooth and stable lane changes with a 100% success rate in diverse driving situations simulated in the study.

Zou et al. [45] proposed a method that combined DDPG and imitation learning (DDPG-IL) to overcome the limitations of traditional DDPG algorithms. The architecture collected training data (through IL) and stored it in an expert pool first. The algorithm then utilized a combination of the demonstration data and its own exploration data for learning. This helped the algorithm converge faster to a stable state. Experimental comparisons conducted on the TORCS racing simulator demonstrate that the DDPG-IL algorithm outperformed the traditional DDPG algorithm.

Zhang et al. [46] proposed a safe and efficient car-following strategy for autonomous driving vehicles based on DDPG algorithm. To ensure safety, the constraint was meant to limit the acceleration of the following vehicle. The physical constraint was also included to prevent abrupt changes in acceleration. Moreover, with the consideration of factors such as jerks, speed errors, and more, the reward function was designed with the aim of enhancing passenger comfort, promoting stability in traffic flow, and improving overall traffic efficiency. Different car trajectories were extracted from the NGSIM dataset for training and testing the model's performance. The results showed the usage of DDPG algorithm along with multiple constraints can easily improve the efficiency of the autonomous vehicle and prevent collisions. Future works include considering other aspects in the reward functions like fuel efficiency etc and making use of more complex environments.

C. PPO BASED PAPERS

Chen et al. [47] used (PPO) with Pure Pursuit (PP) for the vehicle controller architecture. The PP method was used to get a steering control command while the PPO algorithm was used to create a correction command that kept a check on the inaccuracies resulting in a more robust and adaptive controller. This method resulted in the vehicle following the target path more accurately for different speeds and trajectories.

Folkers et al. [48] presented a design approach for a non-linear controller by using the PPO algorithm and defining the policy as a neural network. The training was done on a simulator with obstacles and a more complex environment. The model was tested on a full-sized research vehicle. The controller could handle unknown obstacles and sharp turns by performing evasive manoeuvring.

Ye et al. [49] used the SUMO simulator with a two lane implementation to train the vehicle using safe PPO for lane

changing, overtaking, slowing down and stopping in traffic. They took the state space from all surrounding vehicles around the main agent to train the main algorithm for actions like braking, changing lanes and slowing with respect to the states of the vehicles around it. They accounted for factors like efficiency, safety and comfort in their reward function to provide a smoother training trajectory. As a result, they are able to achieve 95-99% success rate in dense traffic with an average collision rate of 0.5%.

Chen et al. [50] utilised several intelligent driver model (IDM) parameters to take into consideration various driving styles such as aggressive, normal and defensive. PPO with self attention is used to train the main agent. Their model provided better results than DQN and baseline PPO implementation. Their model gave best results with defensive and interactive vehicles while worse but acceptable results with aggressive vehicles. They also observed a decrease in performance proportional to increasing complexity in the environment.

Wu et al. [51] developed an approach based on PPO in which they combine a curiosity-driven technique called random network distillation (RND) to produce an intrinsic reward signal that motivates the agent to investigate its surroundings more effectively. In order to avoid overestimation bias, the evaluation network with smaller predicted values was chosen in the paper's auxiliary critic network, which was added to the original actor-critic framework. The technique was tested in the TORCS driving simulator for the lane-keeping task and the overtaking task and compared to existing DRL techniques. The findings of the trial demonstrated that the strategy they suggested might increase the effectiveness of training and control performance in driving tasks.

Coad et al. [52] proposed a model-free RL based PPO technique for trajectory planning in self-driving cars. The training was done on a Tesla K80 GPU with 32 parallelized environments using the above-mentioned algorithm. Using PPO resulted in a smoother and more efficient path. It also led to less latency to query for the next trajectory. Due to the presence of safety constraints, the vehicle could avoid collisions with obstacles by initializing an emergency stopping module. Future work included using IRL to learn the reward function weights.

Based on this review, we identified a lack of literature that combined lane-keeping and obstacle detection for multiple agents to bring forward a more comprehensive solution to improve autonomous driving. Furthermore, there is a lack of comparison of such algorithms for this specific problem and environment. Hence, the comparative analysis of DDPG and PPO algorithms in terms of training as well as performance analysis is investigated.

IV. AN EMPIRICAL STUDY OF DDPG AND PPO ALGORITHMS

The section discusses the RL model, various scenarios considered for implementation, and a comparative performance analysis of the two algorithms.

TORCS, SUMO [53], and CARLA [54] are the commonly used open-source simulators. SUMO is preferred for large networks. TORCS and CARLA were selected as both represented unique characteristics and use cases. TORCS is a racing simulator that provides racing tracks and other racing cars as traffic. CARLA is an urban simulator with intersections, buildings, traffic signals, civilian traffic and civilian cars.TORCS being a simpler simulator in which artificial intelligence support and training are provided by extra plugins such as gym and server-client connections. CARLA is more complex and is more graphically demanding. It has a python interface that makes it easy to use directly after installation. After trying and testing out both simulators, TORCS was deemed better for the purposes and scope of the research. There are some implementations that use a modified version of the above mentioned simulators. Some papers also have custom 3D environments using Unity and Unreal gaming engine. Furthermore, gym environment gives a more simplistic 2D environment popularly used for double lane problems.

A. REINFORCEMENT LEARNING ELEMENTS

As mentioned in Section II, an RL problem is designed as an MDP. The state space, an action space, and rewards corresponding to favorable state-action pairs for a certain time step are given below.

1) STATE SPACE

The state space for the TORCS environment typically consists of several variables that describe the current state of the race and the car being controlled. Various sensor readings contribute to the state space and include the velocity of the car, the angle between the car and the track axis, the distance between the center of the lane and the car, the distance between the track edge and the car as well as rotation speed of the wheels as shown in Table 1. For the purpose of obstacle and collision avoidance 'opponents' and 'collision' sensors are used. These sensors provide information on the distance of the closest opponents with a range of 200 m and the current damage of the car (the higher the value, the more damage to the vehicle).

Damage sensors in TORCS offer information about the state of various automotive components such as the engine, tires, etc. This sensor monitors many aspects of the vehicle and provides feedback to the operator or the DRL algorithm that controls the vehicle in case any collision takes place. This information can then be used by the algorithm to make decisions on how to adapt its driving strategy or by the user to assess the state of the car and adjust their racing tactics accordingly.

2) ACTION SPACE

Agent's action space is represented in the form of an array a=(Steering, Acceleration, Brake) and is depicted in Table 5.

TABLE 1. Table of different states.

Sensor name	Range	Description
angle(θ)	[-π,π](rad)	Angle between the car and track axis
speedx	(-∞,+∞)(km/hr)	Velocity of the car on the track
track	[0,200]m	Vector of 19 sensors which is used to find the distance of the car and the track edge within 200m
trackPos	(-∞,+∞)	Distance between the car and the track axis
opponent	[0,200]m	Vector of 36 sensors used to detect obstacles
damage	[0,+∞](point)	Shows the current damage value of the car

- An Acceleration value of 1 means that the agent has completely pressed the accelerator pedal and a value of 0 means the agent has completely let go of the accelerator.
- Similarly, the brake value of 1 means full brake applied and 0 is that the brake pedal is not pressed.
- Steering value of -1 means that the car is on the right side and 1 represents that the car is on the left side.

3) REWARDS

Designing appropriate rewards ensure that the agent knows its priorities well which enables the agent to drive safely and get trained faster. The ultimate purpose of the agent is to maximize the rewards. For lane-keeping implementation, the reward function is as shown in Equation 6

$$R = vcos(\theta) - vsin(\theta) - v(\delta) \tag{6}$$

where v = velocity of the car, θ = angle between the car and the track axis, δ = also termed as offset is the distance between the car and the track center.

While implementing collision avoidance, a damage sensor comes into play. The reward function as such does not change, and only a penalty will be applied in case the damage sensor gets activated. If the difference between the current and previous damage sensor values is greater than 0, a collision is considered to have occurred. And in this case, the reward function will change into Equation 7

$$R' = vcos(\theta) - vsin(\theta) - v(\delta) - 50 \tag{7}$$

$$R' = R - 50 \tag{8}$$

B. LANE KEEPING IMPLEMENTATION

TrackPos sensor and angle sensor are used for this purpose. The trackPos sensor provides us with the distance between the car and the track axis. Its value is then normalized between -1 and 1. In case the car(agent) is on the left side of the track axis this sensor outputs value 1, if it's on the right side then the sensor outputs -1. If the car is on the center of the track axis then this sensor outputs a value of 0.

Initially when the execution starts these sensor readings are obtained from the server car and analyzed. The termination conditions are first checked. These conditions include if the agent is out of track, or if the agent has reversed its direction, and lastly, if the agent is stuck or making no progress. If at least one of the above-mentioned termination conditions are satisfied then the process is terminated. If none of the termination conditions are true, we feed the sensor readings to our DRL algorithm to obtain an action and calculate the reward using the already defined reward function (R=vcos(θ)-vsin(θ)-v(trackPos)).

The reward function is designed in such a way that the agent always tries to position itself in the center of the lane. In case the agent starts to go off-track then θ and trackPos values start to increase. This will cause sin(θ) to increase which in turn will cause cos(θ) value to decrease thereby resulting in a negative reward value. This will cause the agent to steer back to the center of the lane and avoid going out of the track.

C. OBSTACLE DETECTION AND AUTO BRAKING

When driving in the lane and if you suddenly encounter an obstacle, the first thing that comes into your mind is applying the brakes and stopping. Opponent sensors are used to facilitate this feature in an autonomous agent. Four of the 36 vector opponent sensors are present in front of the car and are used for detecting front obstacles, the remaining sensors are used for detecting side and rear obstacles. When the simulation starts and before the action is performed by the agent the opponent sensor readings are analyzed.

- If any front obstacle is present within 50 to 20 m, acceleration is made zero so that the vehicle slows down due to the presence of friction between the road and the wheels and ultimately comes to a halt.
- If the distance is less than 20 then brakes are applied with a value of 0.7. This ensures that the vehicle's speed is reduced in a faster manner in order to avoid any collision.
- For side obstacles, if the distance between the obstacle and agent is less than 10 then we apply complete brakes (i.e. value 1.0) so that the autonomous vehicle comes to a stop.

D. COLLISION AVOIDANCE AND OVERTAKING

A multi-agent environment is used for the implementation of these two scenarios. The interaction of multiple agents in an environment is termed as a multi-agent environment. Training in this environment results in the DRL model being more scalable and robust. For multi-agent training, the algorithm gets feedback on a collision with another vehicle through the damage sensor. The higher the damage from the collision, the greater the values received from the damage sensor. Here, however, the exact value of the damage is not required since our objective is no collision. If the difference between the current and previous damage sensor value readings is greater than 0, a penalty is imposed on the reward function which was shown in Equation 7.

With the help of this newly generated reward, the agent realizes the need to avoid collisions in order to obtain

TABLE 2. Summary of the related literature.

General Papers		
Author	Algorithm	Work Summary
Chae et al. [8]	DQN	Proposed a new autonomous braking system based on DRL method (DQN)
Xia et al. [9]	Q-Learning	A new control strategy that combines the experience of professional drivers with a Q-learning algorithm that uses filtered experience replay.
Zhang et al. [18]	DDQN	Proposed DDQN to build the vehicle speed control model to deal with the limitation of Q-learning
Li et al. [19]	DQN	Three-accident prone intersection scenarios were designed on Carla and the algorithm trained on it. Found better than Monte-Carlo based and Bayesian network based methods.
Kartikeyan et al. [20]	DQN	A novel method for intersection navigation consisting of a brake-safe control model and an intersection controller.
Sallab et al. [21]	DDPG, DQN	Compared two DRL methods DDPG and DQN for end-end autonomous driving model (Lane Keeping Assistance)
He et al. [22]	DCPER-DDPG	Proposed an improved version of the DDPG algorithm using double critic networks and a priority experience replay mechanism.
Wang et al. [23]	DQN, DPPG	Proposed two control strategies using different DRL algorithms for lane-keeping assistance modeled on Simulink.
Sharma et al. [11]	Neural Networks	Proposed a DL technique for implementing both lateral and longitudinal control in autonomous vehicles. TORCS simulator was used for training and testing the implementation.
Radwan et al. [24]	PPO, Soft Actor-Crtitc (SAC)	Proposed a paper to obtain the best algorithm in order for the autonomous vehicle to avoid obstacles in a 3D environment obtained through Utility game engine.
Li et al. [10]	DRL algos.	proposed a framework for autonomous vehicles to make decisions in lane changing scenarios using DRL on CARLA simulator
Elallid et al. [25]	DQN	Proposed a reinforcement learning based model using DQN to control the autonomous vehicles in complex scenarios involving pedestrians and moving vehicles using CARLA simulator
Sallab et al. [26]	DQN, DDAC combined with RNN	Proposed a framework that incorporated Recurrent Neural Networks along with DRL algorithms(DQN and DDAC) enabling the car to handle partially observable scenarios
Alzubaidi et al. [27]	DQN	Proposed an EMVs aware LCD model using DQN
Fayjie et al. [28]	DQN	Proposed a DRL based approach to facilitate navigation and avoid obstacles in an urban environment.
Ye et al. [29]	DRL algos.	Proposed a DRL algorithm with continuous action horizon with a combination of off-policy and on-policy.
Huang et al. [30]	RL with DNN	Proposed training an RL algorithm on a DNN using prior human expert demonstrations to reduce training time and increase efficiency.
Baheri et al. [31]	DDQN	Presented a safe DRL based system, combining DDQN with handcrafted and dynamic safety modules.
Guan et al. [32]	MA-PPO	A centralized coordination scheme of autonomous vehicles at intersections without traffic signals using model accelerated PPO with virtual samples to improve on learning efficiency.
Holen et al. [33]	MobileNet with PPO	Gave a combination of DRL algorithm (PPO) with road detection algorithm(MobileNet) to reduce collision rates and improve training efficiency of the algorithm.
Li et al. [34]	MTL-RL	Proposed a method called MTL-RL that utilizes a MTL neural network to take a driver-view image as input, and predict track features along with DRL training.
Cai et al. [35]	DIRL	Proposed a DIRL method with Reveries-Net, an uncertainty aware deep network.

TABLE 2. *(Continued.)* **Summary of the related literature.**

Chen et al. [36]	DRL with sensor input	An interpretabe DRL method that takes camera and lidar input combined and takes decisions based on that.
Mushtaq et al. [37]	DQN	Two stage traffic regulation using a flexible traffic control system and rerouting to divert traffic.
Fernando et al. [38]	Review	A survey of current methods for IRL and evaluation of some popular methods.
Aoki et al. [39]	DQN	A cooperative perception scheme with a custom simulated environment using SUMO traffic and CARLA vehicle.
Hoel et al. [40]	MTCS-NN	Proposed a combined method that uses MTCS and neural networks for planning and tactical decision making based on AlphaGo Zero algorithm.

TABLE 3. **Summary of the DDPG-related literature.**

DDPG Based Papers	
Huang et al. [41]	DDPG algorithm implementation of a single agent for lane keeping scenario on TORCS
Gongsheng et al. [42]	Proposed a vehicle control algorithm designed on a TORCS simulator combined with DDPG. This method made the car more steady when driving and demonstrated excellent learning in less training time.
H.Yi [43]	Multi-agent based DDPG algorithm for autonomous driving to control the continuous actions of vehicles
Wang et al. [44]	Techniques to overcome limitations of lane changing behavior with continuous action in a model-free dynamic driving environment based on DDPG to ensure safety and comfort.
Zou et al. [45]	Proposed a DDPG algorithm based on imitation learning (DDPG-IL) to overcome the limitations of traditional DDPG algorithm.
Zhang et al. [46]	Proposed a safe and efficient car-following strategy for autonomous driving vehicles based on DDPG algorithm. To ensure safety, the safe constraint was meant to limit the acceleration of the following vehicle. The physical constraint was also included to prevent abrupt changes in acceleration. Future works include considering other aspects in the reward functions like fuel efficiency etc and making use of more complex environments.

TABLE 4. **Summary of the PPO related literature.**

PPO Based Papers	
Chen et al. [47]	Used PPO with Pure Pursuit(PP) as the controller. Led in accuracy in following the target path.
Folkers et al. [48]	Trained the agent on a simulator with different obstacles and complex parking environments. Tested the trained algorithm on a research vehicle with good manoeuvring.
Ye et al. [49]	Used SUMO simulator to train agent for lane changing by taking state inputs of surrounding vehicles and accordingly taking action.
Chen et al. [50]	Lane changing and overtaking with different vehicle behaviours (Aggressive, normal and defensive). Compared with DQN and baseline PPO.
Wu et al. [51]	Introduced random network distillation(RND) with PPO to encourage the agent to explore the environment more leading to increased exploration efficiency.
Coad et al. [52]	Proposed a model-free RL based PPO technique for trajectory planning in self-driving cars on a Tesla K80 GPU.

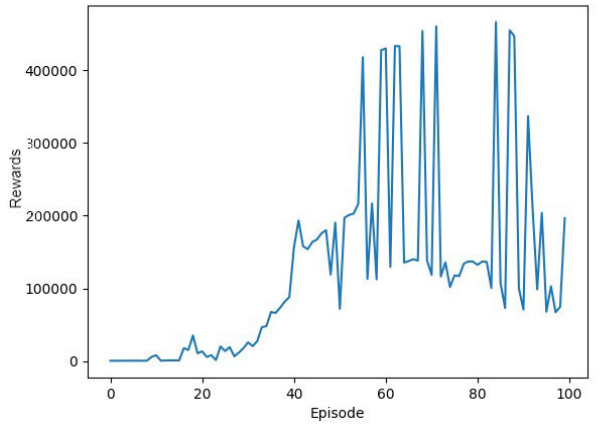
TABLE 5. **Table of actions.**

Action symbol	Description	Range
accelerate	current acceleration	$(0,1) \in R^3$
brake	current braking	$(0,1) \in R^1$
steer	current steering angle	$(-1,1) \in R^1$

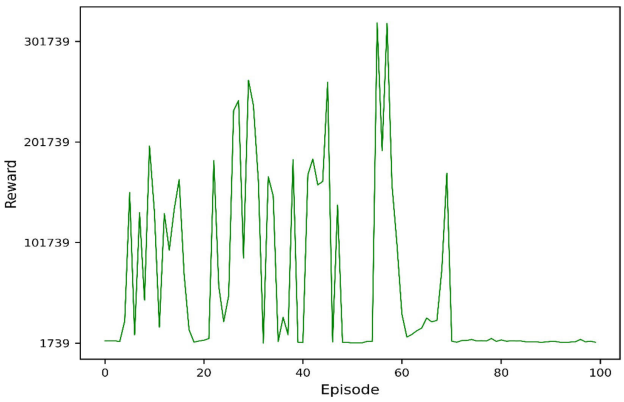
positive rewards throughout the entire duration of the episode. The agent is then continuously trained using the specific algorithms and ultimately learns how to avoid collision by overtaking the vehicles or stopping at a safe distance from them.

E. RESULTS

After extensive training with different hyper-parameter values, the following results provide a comparison between DDPG and PPO. The states and actions taken by an agent from its start to end state are recorded as an episode. Maximum episode length is defined as a maximum number of

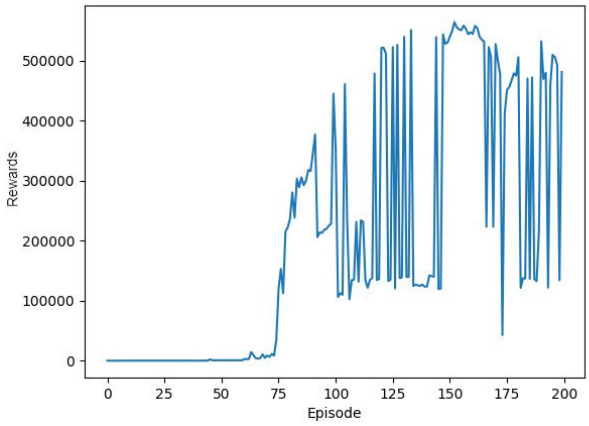


(a) DDPG

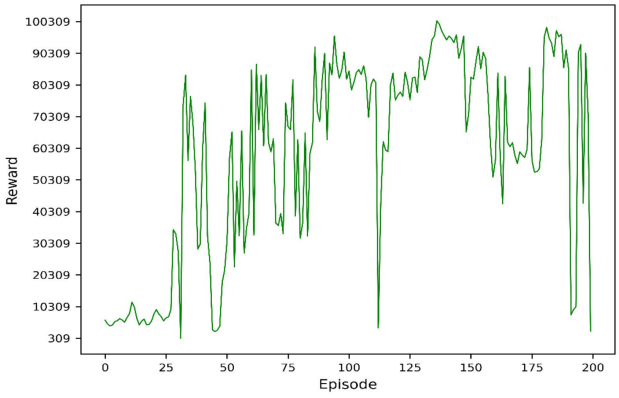


(b) PPO

FIGURE 3. Training results for 100 episodes.



(a) DDPG



(b) PPO

FIGURE 4. Training results for 200 episodes.

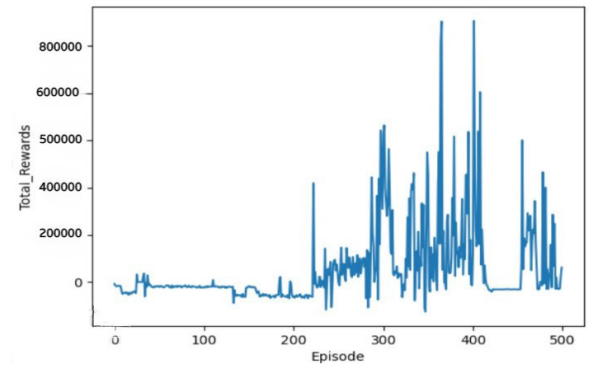
steps for which the algorithm runs. The episode can terminate before it depending on the actions.

Different hyper-parameter values have been taken for both algorithms. The values were chosen after review of implementations of these algorithms as well as hit-and-trial.For DDPG the learning rates and tau for actor and critic networks are 0.001 respectively, For PPO, the learning rates for actor and critic are both 1e-4. As PPO uses clipping, the clipping factor is taken as 0.1.

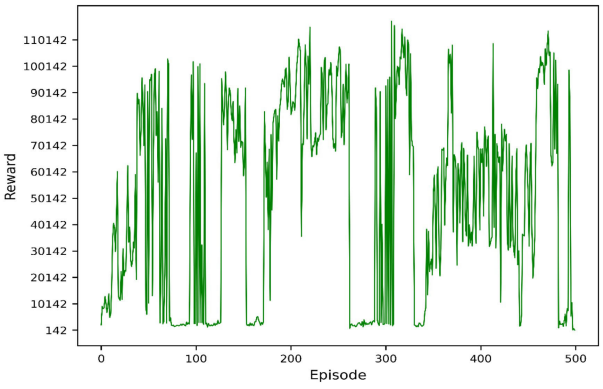
1) SINGLE AGENT

Under the single-agent scenario, the agent is trained under 100,200 and 500 episode durations. The graphs for each of these episode durations are obtained as shown in Figures 3, 4, and 5. It is observed in the beginning that the rewards increase gradually. After some episodes though, the reward might fluctuate between high and low values. This happens because the algorithm tries to optimize between exploration and maximizing rewards. Too little exploration will lead to poor generalization while too much exploration will lead to fewer rewards. Hence, in some episodes, the agent tries to explore a new route by changing its actions which can lead to early episode termination and hence a significant drop in reward values. Low rewards will indicate the agent to get back to its previous action path. Each algorithm has properties that distinguish them and can influence the training results in a specific environment. One of the most prominent differences is DDPG being off-policy as opposed to PPO which is on-policy. This influences how the final policy after a number of episodes will behave. While DDPG selects the best overall policy that gives us the maximum reward, PPO returns the current training policy regardless of its performance compared to previous policies. Hence, in a small number of episodes, PPO may give better results quickly, but it can have a bad policy at the end of the training. In contrast, DDPG steadily learns from experiences and exploration and gives the best policy from the whole training period.

Another difference is the use of experience replay in DDPG and its absence in PPO. Without the replay buffer, the PPO algorithm suffers from a phenomenon called catastrophic forgetting, wherein, because of frequent updates to the policy and a complex environment with a huge number of states, the algorithm starts forgetting previous behaviors and hence starts crashing and giving poorer rewards. This is shown properly in multi-agent training, where the dynamic environment leads to frequent updates of conflicts with previous training and can lead to poorer results. This can be combated by keeping a replay buffer, as done in the DDPG algorithm, where storing the previous trajectories and reusing them can help with better and more stable training. Table 6 below summarizes the above-mentioned findings.



(a) DDPG



(b) PPO

FIGURE 5. Training results for 500 episodes.

TABLE 6. DDPG vs PPO comparison.

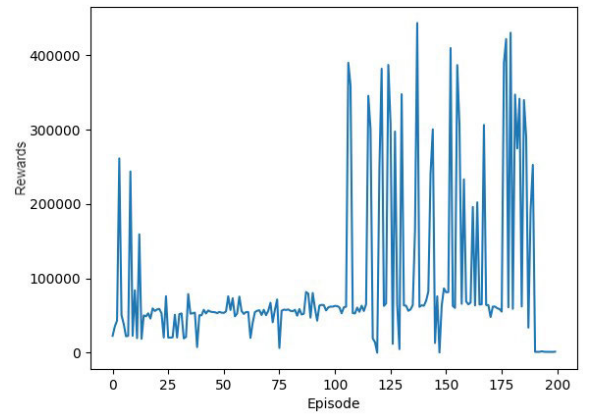
Feature	DDPG	PPO
Training Time	Requires less training time	Requires more training time
Exploration Policy	Off-Policy	On-Policy
Policy Update	Uses a deterministic policy gradient to update the policy	Uses a clipped surrogate objective function to update the policy (Stochastic)
Use of Experience Relay Buffer	Retains experiences and encourages exploration	Does retain old experiences

It can be observed that for all the 3 different episode durations the maximum overall reward obtained by the DDPG is more than that of PPO. For 100 episodes the overall maximum reward for DDPG is 24.5% more than that of PPO. Similarly for 200 and 500 episode durations the maximum reward obtained for DDPG are 79.3% and 86.2% more than that of PPO.

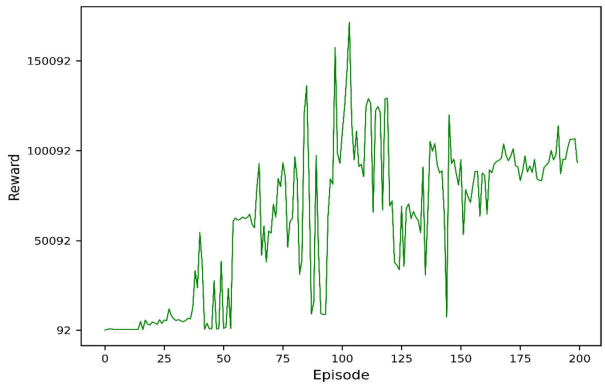
2) MULTI-AGENT

200 episodes of both algorithms were trained on CG track 2 and the rewards were obtained as shown in Figure 6. Here it is observed that DDPG's maximum reward is 64.2% more than that of PPO.

It was observed that in the case of DDPG, the maximum reward obtained during the entire duration of the episode is double that of the PPO. This could be due to the fact that DDPG is an off-policy algorithm that selects the best overall



(a) DDPG



(b) PPO

FIGURE 6. Multi-agent training results for 200 episodes.

policy to obtain the maximum reward while PPO is an on-policy algorithm.

Additionally, DDPG incorporates exploration through the use of noise, which promotes agent discovery of better policies. This exploration capability is beneficial in multi-agent environments where agents need to find effective strategies through trial and error.

V. RESEARCH DIRECTIONS

Even though there is a significant contribution by researchers in the field of autonomous vehicles, the progress is far behind the real-time requirements. Further investigations are necessary to address the challenges in various aspects of the autonomous industry. A few of the research directions are listed below:

- Training the autonomous vehicle for acceptable behavior requires continuous states and actions that increase the dimensionality of the problem and further lead to exponential growth in computational complexity. Thus, designing a DRL-based control system with low overhead to reduce the cost of deployment is really a challenge.
- Most of the research has been conducted in a simulated environment. There are certain limitations to how well the training will account for real-life driving situations. Many physical factors such as friction, air drag, weather

conditions, and road conditions cannot be adequately represented in simulations. Hence, there is a need for a low-cost testbed or an infrastructure that the researcher can use to verify the designed models.

- The multi-agent scenario only takes into account the main agent's sensors for training which is inadequate to have higher accuracy. In the future, the states of all surrounding vehicles for making decisions for lane changing, overtaking, etc., can be considered for better generalization.

VI. CONCLUSION

The concept of autonomous driving has sparked a lot of interest and attention in recent decades because it is thought to provide numerous benefits for individuals and society, including increased road safety, reduced traffic congestion, accidents, and death, and time and pollution savings on commuting. There has been a lot of work and experimentation going on regarding this topic to enable cars to learn their environment, make human-like decisions, and drive autonomously. Reinforcement learning opens up avenues to train the agent to take action in such complex and stochastic environments and generalize it for similar environments.

In the investigation, two different DRL algorithms, DDPG and PPO which can handle high-dimensional state and action spaces have been trained and tested in a simulator for single and multi-agent scenarios. Their results are compared to highlight the differences in how the algorithms work and how efficient they are for this particular problem and environment. Certain characteristics of each algorithm play to their advantage or disadvantage when they are trained in a complex road environment with many states and action pairs at each step and larger training periods. A simple multi-agent implementation is done with the addition of a collision sensor to the other sensors used for a single agent.

Furthermore, a comprehensive literature survey covers the different algorithms and novel methods researched in the past few years in different problem areas related to autonomous driving such as intersections, parking, stationary obstacle avoidance, etc. Research for this problem area using DDPG and PPO has also been covered separately to give a general overview of the past usage of these algorithms specific to autonomous driving.

ACKNOWLEDGMENT

(Sanjna Siboo, Anushka Bhattacharyya, Rashmi Naveen Raj, and S. H. Ashwin contributed equally to this work.)

REFERENCES

[1] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4909–4926, Jun. 2022.

[2] *Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems*, SAE, J3016, 2021.

[3] S. Singh and B. S. Saini, "Autonomous cars: Recent developments, challenges, and possible solutions," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 1022, no. 1, Jan. 2021, Art. no. 012028.

[4] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, "A survey of deep learning applications to autonomous vehicle control," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 2, pp. 712–733, Feb. 2021.

[5] Z. Yao, X. Liang, G.-P. Jiang, and J. Yao, "Model-based reinforcement learning control of electrohydraulic position servo systems," *IEEE/ASME Trans. Mechatronics*, vol. 28, no. 3, pp. 1446–1455, Jun. 2023.

[6] R. N. Raj, A. Nayak, and M. S. Kumar, "A survey and performance evaluation of reinforcement learning based spectrum aware routing in cognitive radio ad hoc networks," *Int. J. Wireless Inf. Netw.*, vol. 27, no. 1, pp. 144–163, Mar. 2020.

[7] Z. Zhu and H. Zhao, "A survey of deep RL and IL for autonomous driving policy learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 14043–14065, Sep. 2022.

[8] H. Chae, C. M. Kang, B. Kim, J. Kim, C. C. Chung, and J. W. Choi, "Autonomous braking system via deep reinforcement learning," in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2017, pp. 1–6.

[9] W. Xia, H. Li, and B. Li, "A control strategy of autonomous vehicles based on deep reinforcement learning," in *Proc. 9th Int. Symp. Comput. Intell. Design (ISCID)*, vol. 2, Dec. 2016, pp. 198–201.

[10] 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," *Transp. Res. C, Emerg. Technol.*, vol. 134, Jan. 2022, Art. no. 103452.

[11] S. Sharma, G. Tewolde, and J. Kwon, "Lateral and longitudinal motion control of autonomous vehicles using deep learning," in *Proc. IEEE Int. Conf. Electro Inf. Technol. (EIT)*, May 2019, pp. 1–5.

[12] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions," *J. Big Data*, vol. 8, no. 1, pp. 1–74, Mar. 2021.

[13] I. H. Sarker, "Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions," *Social Netw. Comput. Sci.*, vol. 2, no. 6, Nov. 2021, Art. no. 420.

[14] J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A theoretical analysis of deep Q-learning," in *Proc. 2nd Conf. Learn. Dyn. Control*, 2020, pp. 486–489.

[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[16] S. H. Ashwin and R. N. Raj, "Deep reinforcement learning for autonomous vehicles: Lane keep and overtaking scenarios with collision avoidance," *Int. J. Inf. Technol.*, vol. 15, no. 7, pp. 3541–3553, Oct. 2023.

[17] G.-P. Antonio and C. Maria-Dolores, "AIM5LA: A latency-aware deep reinforcement learning-based autonomous intersection management system for 5G communication networks," *Sensors*, vol. 22, no. 6, p. 2217, Mar. 2022.

[18] Y. Zhang, P. Sun, Y. Yin, L. Lin, and X. Wang, "Human-like autonomous vehicle speed control by deep reinforcement learning with double Q-learning," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2018, pp. 1251–1256.

[19] G. Li, S. Li, S. Li, Y. Qin, D. Cao, X. Qu, and B. Cheng, "Deep reinforcement learning enabled decision-making for autonomous driving at intersections," *Automot. Innov.*, vol. 3, no. 4, pp. 374–385, Dec. 2020.

[20] P. Karthikeyan, W.-L. Chen, and P.-A. Hsiung, "Autonomous intersection management by using reinforcement learning," *Algorithms*, vol. 15, no. 9, p. 326, Sep. 2022.

[21] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "End-to-end deep reinforcement learning for lane keeping assist," 2016, *arXiv:1612.04340*.

[22] R. He, H. Lv, S. Zhang, D. Zhang, and H. Zhang, "Lane following method based on improved DDPG algorithm," *Sensors*, vol. 21, no. 14, p. 4827, Jul. 2021.

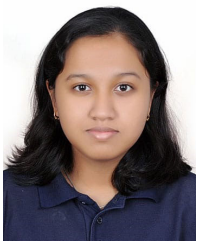
[23] Q. Wang, W. Zhuang, L. Wang, and F. Ju, "Lane keeping assist for an autonomous vehicle based on deep reinforcement learning," SAE Tech. Paper 2020-01-0728, 2020.

[24] M. O. Radwan, A. A. H. Sedky, and K. M. Mahar, "Obstacles avoidance of self-driving vehicle using deep reinforcement learning," in *Proc. 31st Int. Conf. Comput. Theory Appl. (ICCTA)*, Dec. 2021, pp. 215–222.

[25] B. B. Elallid, N. Benamar, N. Mrani, and T. Rachidi, "DQN-based reinforcement learning for vehicle control of autonomous vehicles interacting with pedestrians," in *Proc. Int. Conf. Innov. Intell. Informat., Comput., Technol. (3ICT)*, Nov. 2022, pp. 489–493.

[26] A. El Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," 2017, *arXiv:1704.02532*.

[27] A. Alzubaidi, A. S. A. Sumaiti, Y.-J. Byon, and K. A. Hosani, "Emergency vehicle aware lane change decision model for autonomous vehicles using deep reinforcement learning," *IEEE Access*, vol. 11, pp. 27127–27137, 2023.
[28] A. R. Fayjie, S. Hossain, D. Oualid, and D.-J. Lee, "Driverless car: Autonomous driving using deep reinforcement learning in urban environment," in *Proc. 15th Int. Conf. Ubiquitous Robots (UR)*, Jun. 2018, pp. 896–901.
[29] X. Ye, X. Tang, and K. Yu, "Decision-making for autonomous vehicles on highway: Deep reinforcement learning with continuous action horizon," 2022, *arXiv:2008.11852*.
[30] Z. Huang, J. Wu, and C. Lv, "Efficient deep reinforcement learning with imitative expert priors for autonomous driving," 2021, *arXiv:2103.10690*.
[31] A. Baheri, S. Nageshrao, H. E. Tseng, I. Kolmanovsky, A. Girard, and D. Filev, "Deep reinforcement learning with enhanced safety for autonomous highway driving," 2020, *arXiv:1910.12905*.
[32] Y. Guan, Y. Ren, S. E. Li, Q. Sun, L. Luo, and K. Li, "Centralized cooperation for connected and automated vehicles at intersections by proximal policy optimization," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 12597–12608, Nov. 2020.
[33] M. Holen, R. Saha, M. Goodwin, C. W. Omlin, and K. E. Sandsmark, "Road detection for reinforcement learning based autonomous car," in *Proc. 3rd Int. Conf. Inf. Sci. Syst.* New York, NY, USA: Association for Computing Machinery, Mar. 2020, pp. 67–71.
[34] D. Li, D. Zhao, Q. Zhang, and Y. Chen, "Reinforcement learning and deep learning based lateral control for autonomous driving [application notes]," *IEEE Comput. Intell. Mag.*, vol. 14, no. 2, pp. 83–98, May 2019.
[35] P. Cai, H. Wang, H. Huang, Y. Liu, and M. Liu, "Vision-based autonomous car racing using deep imitative reinforcement learning," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 7262–7269, Oct. 2021.
[36] J. Chen, S. E. Li, and M. Tomizuka, "Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 5068–5078, Jun. 2022.
[37] A. Mushtaq, I. U. Haq, M. U. Imtiaz, A. Khan, and O. Shafiq, "Traffic flow management of autonomous vehicles using deep reinforcement learning and smart rerouting," *IEEE Access*, vol. 9, pp. 51005–51019, 2021.
[38] T. Fernando, S. Denman, S. Sridharan, and C. Fookes, "Deep inverse reinforcement learning for behavior prediction in autonomous driving: Accurate forecasts of vehicle motion," *IEEE Signal Process. Mag.*, vol. 38, no. 1, pp. 87–96, Jan. 2021.
[39] S. Aoki, T. Higuchi, and O. Altintas, "Cooperative perception with deep reinforcement learning for connected vehicles," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Oct. 2020, pp. 328–334.
[40] C.-J. Hoel, K. Driggs-Campbell, K. Wolff, L. Laine, and M. J. Kochenderfer, "Combining planning and deep reinforcement learning in tactical decision making for autonomous driving," *IEEE Trans. Intell. Vehicles*, vol. 5, no. 2, pp. 294–305, Jun. 2020.
[41] Z. Huang, J. Zhang, R. Tian, and Y. Zhang, "End-to-end autonomous driving decision based on deep reinforcement learning," in *Proc. 5th Int. Conf. Control, Autom. Robot. (ICCAR)*, Apr. 2019, pp. 658–662.
[42] Z. Gongsheng, P. Chunmei, D. Jiang, and S. Junfeng, "Deep deterministic policy gradient algorithm based lateral and longitudinal control for autonomous driving," in *Proc. 5th Int. Conf. Mech., Control Comput. Eng. (ICMCCE)*, Dec. 2020, pp. 740–745.
[43] H. Yi, "Deep deterministic policy gradient for autonomous vehicle driving," in *Proc. Int. Conf. Artif. Intell. (ICAI)*. Athens, Greece: The Steering Committee the World Congress in Computer Science, 2018, pp. 191–194.
[44] P. Wang, H. Li, and C.-Y. Chan, "Continuous control for automated lane change behavior based on deep deterministic policy gradient algorithm," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2019, pp. 1454–1460.
[45] Q. Zou, K. Xiong, and Y. Hou, "An end-to-end learning of driving strategies based on DDPG and imitation learning," in *Proc. Chin. Control Decis. Conf. (CCDC)*, Aug. 2020, pp. 3190–3195.
[46] Y. Zhang and R. Yan, "Safe car-following strategy with multi-constraints based on deep reinforcement learning for autonomous driving vehicles," in *Proc. IEEE Int. Conf. Unmanned Syst. (ICUS)*, Oct. 2022, pp. 1151–1156.
[47] I.-M. Chen and C.-Y. Chan, "Deep reinforcement learning based path tracking controller for autonomous vehicle," *Proc. Inst. Mech. Eng. D, J. Automobile Eng.*, vol. 235, nos. 2–3, pp. 541–551, Feb. 2021.
[48] A. Folkers, M. Rick, and C. Büskens, "Controlling an autonomous vehicle with deep reinforcement learning," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2019, pp. 2025–2031.
[49] Y. Fei, X. Cheng, P. Wang, C. Chan, and J. Zhang, "Automated lane change strategy using proximal policy optimization-based deep reinforcement learning," 2020, *arXiv:2002.02667*.
[50] X. Chen, J. Wei, X. Ren, K. H. Johansson, and X. Wang, "Automatic overtaking on two-way roads with vehicle interactions based on proximal policy optimization," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jul. 2021, pp. 1057–1064.
[51] Y. Wu, S. Liao, X. Liu, Z. Li, and R. Lu, "Deep reinforcement learning on autonomous driving policy with auxiliary critic network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 7, pp. 3680–3690, Jul. 2021.
[52] J. Coad, Z. Qiao, and J. M. Dolan, "Safe trajectory planning using reinforcement learning for self driving," 2020, *arXiv:2011.04702*.
[53] 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 *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 2575–2582.
[54] A. Dosovitskiy, G. Ros, F. Codevilla, M. Antonio, and V. Koltun, "CARLA: An open urban driving simulator," *CoRR*, vol. abs/1711.03938, pp. 1–16, Nov. 2017.



SANJNA SIBOO is currently pursuing the B.Tech. degree in computer and communication engineering with a minor specialization in computational intelligence with the Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, Karnataka, India.

She is a Software Developer with Microsoft, Hyderabad, India. Her research interests include cloud computing, artificial intelligence, and machine learning.

ANUSHKA BHATTACHARYYA is currently pursuing the B.Tech. degree in computer and communication engineering with a minor specialization in big data with the Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, Karnataka, India.

She is a Technical Associate with the GSK Global Capability Centre, Bengaluru, India. Her research interests include machine learning, data science, and big data.

RASHMI NAVEEN RAJ (Member, IEEE) received the B.E. degree in electrical and electronics engineering from Mangalore University, Karnataka, India, in 2001, and the M.Tech. degree in digital electronics and advanced communication and the Ph.D. degree in cognitive radio ad hoc networks from the Manipal Academy of Higher Education, Manipal, Karnataka, India, in 2009 and 2021, respectively.

She is currently an Associate Professor with the Department of Information and Communication Technology, Manipal Institute of Technology, Manipal Academy of Higher Education. She has 17 years of teaching experience to engineering students with the National Institute of Technology Karnataka and the Manipal Institute of Technology. She has published many articles in good journals. Her research interests include cognitive radio networks, machine learning, and autonomous driving vehicles.

S. H. ASHWIN received the B.Tech. degree in computer and communication engineering from the Manipal Institute of Technology, MAHE, Manipal, Karnataka, India, in 2022, with a minor specialization in computational mathematics. He is a Development Engineer with Standard Chartered Global Business Services, Bengaluru, India. His research interests include reinforcement learning and cyber security.