IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 12, May 2023

Automated Vehicle in CARLA

Saurabh Chaudhari[1], Vatsal Mehta[2], Vishvam Choure[3]
Students, Department of Computer Engineering[1,2,3]
Sinhgad Institute of Technology, Lonavala, Maharashtra, India

Abstract: *The advent of automated driving technologies has brought about a paradigm shift in the automotive industry. The development and evaluation of automated vehicles (AVs) play a crucial role in shaping the future of transportation. To ensure the safe and efficient deployment of autonomous vehicles on public roads, rigorous testing and validation processes are essential. However, conducting real-world tests can be costly, time-consuming, and pose potential risks. To address these challenges, simulation environments such as the CARLA (Car Learning to Act) simulator have emerged as valuable tools for testing automated vehicles. This paper presents scenarios conducted within the CARLA simulator, focusing on critical aspects of automated driving, such as perception, decision-making, and control. The results obtained from the CARLA simulator experiments provide valuable insights into the strengths and limitations of the tested automated driving systems.*

Keywords: Automated vehicle, CARLA, Simulator, Automated driving

I. INTRODUCTION

The development and widespread adoption of automated vehicles (AVs) have the potential to revolutionize transportation systems, offering increased safety, efficiency, and convenience. However, the complexity and inherent risks associated with AV technology necessitate extensive testing and evaluation before their deployment on real roads. Conducting large-scale real-world experiments can be time-consuming, expensive, and often limited by safety considerations. To overcome these challenges, researchers increasingly turn to simulation environments to develop, refine, and evaluate AV systems.One such prominent simulation platform is the CARLA Simulator, an open-source software framework that provides a realistic and customizable environment for AV research. CARLA offers high-fidelity graphics, accurate sensor modelling, and sophisticated traffic simulation, enabling researchers to replicate real-world driving scenarios within a controlled virtual environment. By leveraging the capabilities of CARLA, researchers can iterate rapidly, test various algorithms and configurations, and validate the performance of AV systems before transitioning to real-world testing.this research paper explores the integration of machine learning techniques, specifically reinforcement learning, to enhance AV capabilities. By training the AV system within the CARLA Simulator, researchers can expose the AV to diverse driving scenarios, enabling it to learn and adapt its decision-making strategies in complex environments.

II. LITERATURE REVIEW

The research on AVs has gained significant momentum in recent years, driven by the potential of AVs to transform transportation systems worldwide. With the growing interest in AV development and evaluation, simulation platforms have emerged as crucial tools for researchers to test and refine AV technologies. In this literature review, we explore the existing body of research related to AV development and evaluation using the CARLA Simulator. Simulation environments, such as the CARLA Simulator, provide a controlled and safe environment for testing AV systems before conducting real-world experiments. CARLA, in particular, has gained prominence due to its realistic graphics, sensor modelling capabilities, and dynamic traffic simulation. Many researchers have utilized CARLA as a foundation for their AV research, harnessing its features to design and evaluate advanced control systems.

There is a wide range of simulators available for testing autonomous vehicle solutions, each with its own advantages and disadvantages. Examples include CarCraft, Udacity, TORCS, and RRADS, among others. However, these simulators lack the dynamic and complex nature of real-world environments, such as the presence of pedestrians, intersections, traffic rules, and other factors that differentiate urban driving from simple track racing. Additionally,

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 12, May 2023

certain closed-source commercial simulators like Grand Theft Auto V, PreScan, and ANSYS offer limited customization and control over the environment, restricted kinematic behaviour, limited scripting and scenario specifications, constrained sensor suite options, and other limitations due to their commercial nature.

An ideal simulator should strive for maximum realism, encompassing detailed 3D environments and accurate lower-level vehicle calculations that account for the physics of the vehicle. There is always a trade-off between the fidelity of the 3D environment and the simplification of vehicular dynamics. CARLA simulator and LGSVL successfully strikes this balance, positioning themselves as cutting-edge simulators in the field.[1][2]

Classic autonomous driving systems commonly employ advanced sensors for perceiving the environment and complex control algorithms to ensure safe navigation in various challenging scenarios. These systems typically adopt a modular architecture, where individual modules process information in an asynchronous manner. The perception layer acquires data from different sensors like cameras, LiDAR, RADAR, GNSS, IMU, and others to gather information about the surroundings. In terms of the control layer, several widely-used control methods include PID control, Model Predictive Control (MPC) algorithm, Fuzzy Control, Model-Reference Adaptive Control, Fractional Order Control, Pure-Pursuit (PP) path tracking control, and Linear-Quadratic Regulator (LQR) algorithm. Despite exhibiting satisfactory performance, these controllers often rely on the specific environment, necessitating careful fine-tuning of their corresponding hyperparameters to achieve the desired behaviour. This task of hyperparameter adjustment is non-trivial and requires meticulous attention.[3]

III. CARLA SIMULATOR - OVERVIEW

CARLA simulator is a highly regarded and widely used platform for research and development in the field of autonomous driving. It offers a comprehensive and realistic environment for testing and evaluating autonomous vehicle algorithms and systems. This section provides an overview of the CARLA simulator, highlighting its key features and capabilities.

Realistic 3D Environment: CARLA provides a highly realistic and visually immersive 3D environment for simulating autonomous driving scenarios. The simulator incorporates detailed road networks, buildings, and urban elements to closely resemble real-world urban landscapes. This level of realism helps researchers assess the performance and behaviour of their autonomous systems in a variety of challenging and dynamic environments.

High Fidelity Vehicle Dynamics: CARLA excels in modelling the intricate dynamics of vehicles, ensuring that the simulated vehicles accurately mimic real-world behaviour. The simulator takes into account various factors such as acceleration, braking, steering, suspension, and tire physics. This high fidelity in vehicle dynamics enables researchers to evaluate and fine-tune their control algorithms and assess how their autonomous systems handle different driving situations.

Sensor Simulation: CARLA offers comprehensive sensor simulation capabilities, supporting a wide range of sensors commonly used in autonomous vehicles. Researchers can incorporate cameras, LiDAR, RADAR, GPS, and other sensors into their simulated vehicles. These sensors capture data from the environment, allowing researchers to develop and test perception algorithms for object detection, lane detection, and scene understanding.

Customization and Control: CARLA provides researchers with extensive customization options to create diverse and challenging scenarios. Researchers can adjust various parameters such as traffic density, pedestrian behaviour, weather conditions, and road network layout. This level of customization enables the creation of specific scenarios tailored to the research objectives, facilitating thorough evaluation and comparison of different algorithms and systems.

Scenario Generation: CARLA enables the generation of complex and dynamic driving scenarios. Researchers can design scenarios that involve urban intersections, highway merging, overtaking, and other challenging manoeuvres. This capability allows researchers to test and validate their algorithms under various conditions, including interactions with other vehicles, pedestrians, and traffic rules. It provides a valuable means to assess the robustness and adaptability of autonomous driving systems.

Open-Source and Extensibility: CARLA is an open-source simulator, meaning that its source code is freely available for modification and enhancement. This open nature fosters collaboration among researchers and encourages the development of new features and improvements to the simulator. Additionally, CARLA supports integration with

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 12, May 2023

external tools and libraries through well-defined APIs, enabling researchers to leverage additional functionalities and algorithms in their experiments.

IV. PID CONTROL METHOD

The PID (Proportional-Integral-Derivative) controller is a widely used control algorithm for automated vehicles, including those simulated in the CARLA simulator. The PID controller aims to regulate the vehicle's motion by continuously adjusting the control inputs, such as throttle, brake, and steering angle, based on the error between the desired and actual states.

The PID controller consists of three components:

- Proportional (P) Control: The proportional term calculates the control input based on the instantaneous error between the desired and actual states. It amplifies the error and produces a control output that is proportional to this error. In the context of automated driving, the error can represent variables such as lateral deviation from the desired path or heading error. The proportional gain, denoted as Kp, determines the sensitivity of the control response to the error. A higher Kp value results in a stronger response but may lead to overshooting or oscillatory behaviour.
- Integral (I) Control: The integral term takes into account the cumulative error over time. It integrates the error and corrects for any steady-state error or bias that may exist. The integral term is responsible for eliminating long-term errors and providing more robust control. The integral gain, denoted as Ki, determines the responsiveness of the control system to this cumulative error. If the error persists for a longer duration, the integral term becomes larger, resulting in a stronger corrective action.
- Derivative (D) Control: The derivative term anticipates the future behaviour of the error based on its rate of change. It calculates the rate of change of the error and applies a control input to counteract the error's trend. The derivative term provides damping to the control response, helping to reduce overshooting and stabilize the system. The derivative gain, denoted as Kd, determines the responsiveness of the control system to the rate of change of the error. A higher Kd value increases the damping effect but may lead to increased noise sensitivity.

In the CARLA simulator, the PID controller is typically implemented as part of the control module for the automated vehicle. The controller receives inputs from various sensors, such as cameras, LiDAR, or GPS, to estimate the current vehicle state, such as position, velocity, and orientation. It also receives the desired trajectory or path as an input.

Based on the error between the desired trajectory and the estimated vehicle state, the PID controller computes the control inputs for the vehicle, such as throttle, brake, and steering angle. The proportional term adjusts the control input based on the immediate error, the integral term accounts for any steady-state errors or biases, and the derivative term helps anticipate and counteract the error trend.

The gains (Kp, Ki, and Kd) of the PID controller need to be carefully tuned to achieve desired performance and stability. This tuning process involves iterative adjustments to find the appropriate gains that result in smooth, accurate, and stable vehicle control. Different techniques, such as manual tuning or automatic tuning algorithms, can be employed to optimize the PID controller's performance.

The formula for PID controller is as follows:

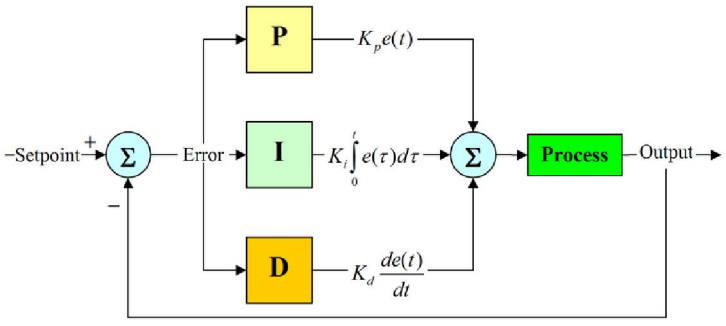

Fig. 1. PID Controller

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 12, May 2023

where,

u(t) = PID control variable
e(t) = error value
Ki = integral gain
dt = change in time
Kp = proportional gain
τ = variable of integration
de = change in error value

V. DEEP REINFORCEMENT LEARNING

Deep reinforcement learning (DRL) is a cutting-edge approach for training automated vehicles in the CARLA simulator. It combines the power of deep learning with reinforcement learning techniques to enable the vehicle to learn optimal decision-making policies directly from raw sensor data.

In the context of automated vehicles, reinforcement learning involves an agent (the vehicle) interacting with an environment (the CARLA simulator) and learning to make sequential decisions to maximize a reward signal. The agent learns through a trial-and-error process, where it explores the environment, takes actions, and receives feedback in the form of rewards or penalties based on the quality of its decisions.

DRL extends reinforcement learning by leveraging deep neural networks to approximate the agent's policy and value functions. These neural networks, commonly known as deep Q-networks (DQNs), enable the agent to handle high-dimensional input data, such as images from cameras or point cloud data from LiDAR sensors.

The DRL process typically involves the following key components:

State Representation: In the CARLA simulator, the state representation for the DRL agent can include various inputs from sensors, such as images, LiDAR point clouds, or vehicle telemetry data. The state captures the relevant information about the vehicle's surroundings, including the position of other vehicles, pedestrians, traffic signs, and road conditions.

Action Space: The action space defines the set of possible actions the DRL agent can take in the CARLA simulator. For an automated vehicle, these actions may include accelerating, braking, and steering. The action space can be discretised or continuous, depending on the specific control requirements.

Reward Design: The reward function provides feedback to the agent based on its actions in the CARLA simulator. The reward function should be carefully designed to encourage desirable behaviour, such as safe driving, staying within lanes, obeying traffic rules, and reaching the destination efficiently. Proper reward shaping is crucial to guide the learning process and achieve desired driving behaviours.

Deep Q-Network (DQN): The DQN is a deep neural network that serves as a function approximator for the agent's policy and value functions. It takes the state representation as input and outputs action values for each possible action in the action space. The DQN is trained to optimize the expected cumulative reward over time using a combination of techniques like experience replay and target networks to stabilize the learning process.

Training Process: The DRL agent learns by iteratively exploring the CARLA simulator, taking actions, and observing the resulting state transitions and rewards. The agent updates the DQN's weights using optimization algorithms such as stochastic gradient descent or variants like Adam. The training process aims to find the optimal policy that maximizes the cumulative reward in the CARLA environment.

Transfer Learning: Transfer learning techniques can be employed to leverage pre-trained models or knowledge gained from similar tasks or environments. By transferring knowledge from pre-training tasks, the DRL agent can accelerate learning in the CARLA simulator and adapt more quickly to new scenarios.

DRL for automated vehicles in the CARLA simulator offers several advantages. It enables the vehicle to learn complex decision-making policies directly from raw sensor data, allowing it to adapt to diverse driving scenarios and learn from real-world-like experiences. DRL can handle high-dimensional inputs and learn representations that capture meaningful features from the environment. Additionally, the flexibility of DRL allows for continuous improvement and adaptation as new data and experiences are collected.

VI. DEEP Q-NETWORK

Deep Q-Network (DQN) is a powerful algorithm for training automated vehicles in the CARLA simulator using deep reinforcement learning. It combines the concepts of Q-learning and deep neural networks to enable the vehicle to learn

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 12, May 2023

optimal driving policies directly from raw sensor data.

The DQN algorithm consists of the following key components:

State Representation: In the CARLA simulator, the state representation for the DQN agent typically includes inputs from various sensors, such as cameras, LiDAR, or GPS. These sensors capture information about the vehicle's surroundings, including the positions of other vehicles, pedestrians, traffic signs, and road conditions. The raw sensor data is used as input to the DQN.

Action Space: The action space defines the set of possible actions that the DQN agent can take in each state. For an automated vehicle, these actions may include accelerating, braking, and steering. The action space can be discretised or continuous, depending on the specific control requirements of the autonomous driving task.

Deep Neural Network: The core component of the DQN algorithm is a deep neural network, commonly known as the Q-network. The Q-network approximates the Q-function, which estimates the expected cumulative reward for taking a particular action in a given state. The input to the Q-network is the state representation, and the output is the Q-values for each action in the action space. The Q-network is typically implemented as a convolutional neural network (CNN) or a combination of convolutional and fully connected layers.

Experience Replay: To enhance the stability and efficiency of learning, DQN incorporates experience replay. Experience replay involves storing the agent's experiences (state, action, reward, next state) in a replay buffer during interactions with the CARLA simulator. During the training process, a random batch of experiences is sampled from the replay buffer, breaking the sequential correlation between experiences. This random sampling helps to mitigate the issues of data correlation and allows for more efficient learning.

Exploration and Exploitation: To balance exploration and exploitation, which is crucial for effective learning, DQN often uses an epsilon-greedy strategy. During training, the agent chooses actions based on a combination of exploration and exploitation. With a certain probability (epsilon), the agent selects a random action to explore the environment. Otherwise, it selects the action with the highest Q-value (exploitation) according to the current Q-network.

The DQN training process involves iteratively interacting with the CARLA simulator, collecting experiences, and updating the Q-network. The objective is to learn the optimal Q-values that maximize the expected cumulative reward over time. The training is typically performed through gradient descent, where the loss function is defined as the mean squared error between the predicted Q-values and the target Q-values.

By training an automated vehicle using DQN in the CARLA simulator, the agent can learn to make optimal driving decisions based on raw sensor data. The DQN algorithm enables the vehicle to adapt to diverse driving scenarios and learn from real-world-like experiences. Through the combination of deep learning and reinforcement learning, DQN can handle high-dimensional input data and learn complex decision-making policies for autonomous navigation.

VII. RESULT

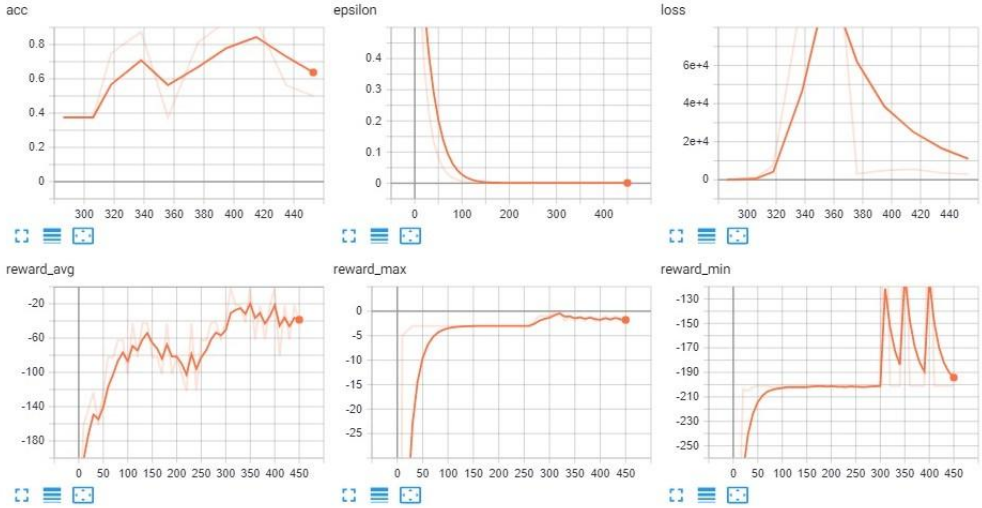

Fig. 2. DQN TensorBoard – 500 Episodes

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 12, May 2023

The results of the implemented DQN algorithm are displayed in the TensorBoard, where performance evaluation is conducted based on specific parameters and presented through graphical representations. By comparing the TensorBoard graphs, it can be deduced that the model's performance is positively correlated with the number of training episodes.

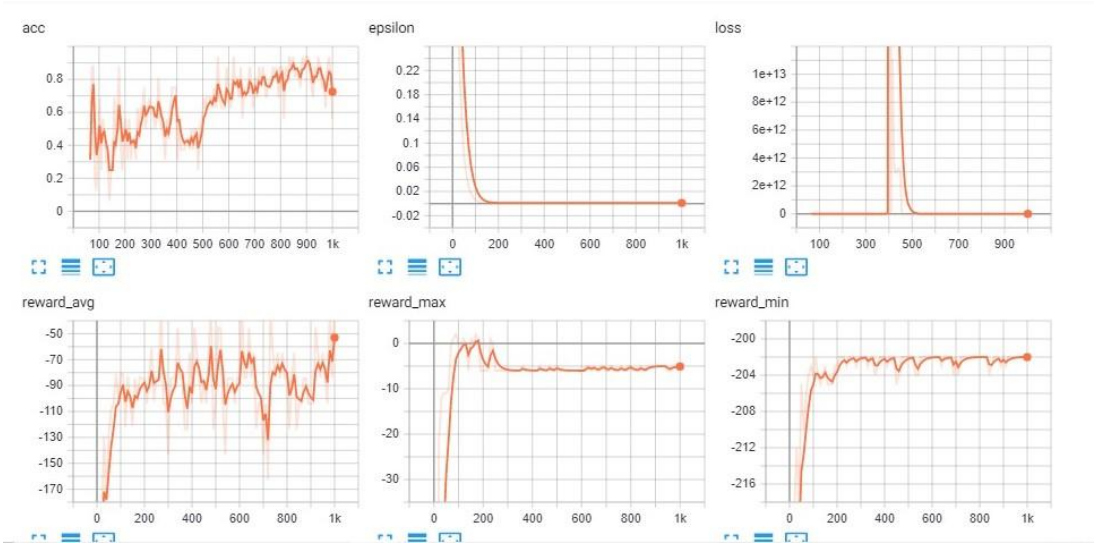

Fig. 3.DQN TensorBoard – 1000 Episodes

VIII. CONCLUSION & FUTURE WORK

This research paper presents two distinct approaches to autonomous driving navigation using the CARLA simulator. The first approach employs the PID control method, integrating advanced sensors for environmental perception and a sophisticated control algorithm to ensure safe navigation. The second approach utilizes DQN, a Deep Reinforcement Learning algorithm, as the basis for autonomous driving navigation. In the future we will investigate ways to optimize the performance of the PID control method by fine-tuning the control parameters. Additionally, explore techniques to enhance the performance of DQN algorithm or employ different neural network architectures.

REFERENCES

[1]. Sumbal Malik, Manzoor Ahmed Khan, Hesham El-Sayed, CARLA: Car Learning to Act — An Inside, Procedia Computer Science, Volume 198, 2022, Pages 742-749, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2021.12.316.

[2]. Gómez-Huélamo, C., Del Egido, J., Bergasa, L.M. et al. Train here, drive there: ROS based end-to-end Autonomous-driving pipeline validation in CARLA simulator using the NHTSA typology. Multimed Tools Appl 81, 4213–4240 (2022). https://doi.org/10.1007/s11042-021-11681-7

[3]. Pérez-Gil, Ó., Barea, R., López-Guillén, E. et al. Deep reinforcement learning based control for Autonomous Vehicles in CARLA. Multimed Tools Appl 81, 3553–3576 (2022). https://doi.org/10.1007/s11042-021-11437-3

[4]. Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533.

[5]. Lillicrap, T. P., Hunt, J. J., Pritzel, A., et al. (2016). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

[6]. Sutton, R. S., and Barto, A. G. (2018). Reinforcement Learning: An Introduction. MIT Press.