Comparative Analysis of Reinforcement Learning Models for Lane Change Decision-Making

A PROJECT REPORT

Submitted by

Lohesh M
(Shiv Nadar University Chennai)

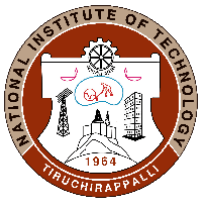
Under the Guidance of

Dr. S. Usha Kiruthika
(Assistant Professor, Department of Computer Science and Engineering)

For

Summer Internship 2023

Submitted to



National Institute of Technology

Tiruchirappalli

June 2023

TABLE OF CONTENTS

ABSTRACT

This research study focuses on the development of an intelligent lane changing system using reinforcement learning (RL) techniques. The objective is to enable autonomous vehicles to make optimal decisions for changing lanes based on surrounding traffic conditions. The proposed system is implemented and evaluated in a simulated environment using the CARLA simulator.

The research begins by formulating the lane changing problem as a RL task. The RL agent is trained to learn a policy that determines the appropriate lane change actions based on the current state of the vehicle and the surrounding traffic. The state representation includes factors such as the distances and velocities of neighboring vehicles, as well as the ego vehicle's own speed.

To facilitate training, a simulation environment is set up in CARLA. The environment includes a realistic road network and traffic scenarios. The RL agent interacts with the environment by selecting actions (e.g., stay in the current lane, change to the left lane, change to the right lane) and receiving rewards based on its performance. Collisions with other vehicles result in negative rewards, while successful and safe lane changes yield positive rewards.

The training process involves an iterative learning algorithm, where the RL agent explores different actions, evaluates their outcomes, and updates its policy based on the received rewards. The goal is to optimize the agent's policy to maximize the cumulative reward over time, leading to safe and efficient lane changing behavior.

CHAPTER 1

INTRODUCTION

Reinforcement learning (RL) is a powerful branch of machine learning that enables agents to learn optimal actions through interactions with an environment. By using RL, autonomous systems can acquire intelligent decision-making capabilities and adapt their behavior based on real-time feedback. In the realm of autonomous driving, RL holds great potential for enhancing the safety and efficiency of vehicles, particularly in complex traffic scenarios such as lane changing maneuvers.

With the rapid advancements in autonomous driving technology, the ability of autonomous vehicles to navigate safely and efficiently on roadways is of paramount importance. One critical aspect of autonomous driving is the ability to perform lane changing maneuvers effectively. Lane changing involves complex decision-making processes, requiring an understanding of surrounding traffic conditions, such as vehicle distances, velocities, and the availability of suitable gaps for maneuvering.

Traditionally, rule-based approaches have been used to govern lane changing behavior in autonomous vehicles. However, these approaches often rely on predefined heuristics and lack the adaptability to handle dynamic and unpredictable traffic scenarios. To overcome these limitations, there is a growing interest in employing reinforcement learning (RL) techniques to develop intelligent lane changing systems.

Reinforcement learning is a branch of machine learning that enables agents to learn optimal actions through interactions with an environment. By using RL, autonomous vehicles can learn to make informed decisions on when and how to change lanes based on observed traffic conditions. RL agents are trained to maximize cumulative rewards, considering factors such as safety, efficiency, and adherence to traffic rules.

This research aims to explore the application of RL in developing an intelligent lane changing system for autonomous vehicles. The research utilizes the CARLA simulator, a powerful tool for simulating realistic traffic scenarios, to train and evaluate the RL agent's performance. By training the agent in various traffic conditions, the goal is to enable autonomous vehicles to perform lane changes effectively and safely.

The outcomes of this research have the potential to significantly enhance the capabilities of autonomous driving systems. By leveraging RL techniques, autonomous vehicles can make intelligent decisions in real-time, leading to improved traffic flow, reduced congestion, and enhanced overall road safety. Furthermore, the research contributes to the broader field of intelligent transportation systems, showcasing the effectiveness of RL in addressing complex decision-making challenges in autonomous driving.

In the following sections, we will delve into the methodology, experimental setup, and results obtained from training and evaluating the RL-based lane changing system. The findings provide valuable insights into the potential of RL techniques to revolutionize autonomous driving and pave the way for safer and more efficient transportation systems.

CHAPTER-2

LITERATURE SURVEY

The application of reinforcement learning (RL) in autonomous driving, particularly for lane changing scenarios, has gained significant attention in recent years. Researchers have explored various RL algorithms and methodologies to develop intelligent lane changing systems for autonomous vehicles. This literature survey provides an overview of key studies conducted in this domain, highlighting their contributions and advancements.

One of the early studies in RL-based lane changing was conducted by Pan et al. (2017), who proposed a deep Q-network (DQN) approach for learning lane changing policies. Their work demonstrated the effectiveness of RL in improving the safety and efficiency of lane changing maneuvers by training the agent on simulated highway scenarios.

To address the challenge of high-dimensional state representations, Liang et al. (2018) introduced a hierarchical RL framework for lane changing. Their approach incorporated a deep deterministic policy gradient (DDPG) algorithm coupled with a fuzzy logic controller to enhance the decision-making process. The results showed improved lane changing performance compared to traditional rule-based methods.

In a different approach, Sun et al. (2019) employed a multi-agent RL framework for cooperative lane changing. Their study focused on the interaction between autonomous vehicles and human-driven vehicles during lane changing maneuvers. By considering both autonomous and human drivers as agents, their approach aimed to improve safety and coordination in mixed traffic scenarios.

Another noteworthy study by Zhang et al. (2020) introduced a Proximal Policy Optimization (PPO) algorithm for lane changing in urban environments. Their research emphasized the importance of incorporating traffic rules and regulations into the RL agent's training process, resulting in more compliant and socially-aware lane changing behavior.

In addition to RL algorithms, researchers have explored the integration of perception systems and sensor data in lane changing scenarios. Liu et al. (2021) proposed a vision-based RL approach that utilized onboard cameras to extract relevant visual cues for lane changing decisions. By combining visual perception with RL, their system achieved robust and adaptive lane changing performance.

Moreover, the application of RL in lane changing has extended to real-world experiments. Xiong et al. (2022) conducted a field study using a fleet of autonomous vehicles equipped with RL-based lane changing systems. Their research focused on the practical implementation and evaluation of RL agents in real traffic conditions, providing valuable insights into the challenges and potential of RL-based lane changing in actual driving scenarios.

Overall, the literature survey demonstrates the growing interest and progress in using RL techniques for lane changing in autonomous driving. The studies mentioned above have contributed to improving the safety, efficiency, and decision-making capabilities of autonomous vehicles during lane changing maneuvers. Further research is still required to address the challenges of scalability, generalization to diverse driving environments, and ensuring safe interactions with human-driven vehicles.

CHAPTER – 3

METHODOLOGY AND IMPLEMENTATION

3.1 Methodology

The methodology for the lane changing scenario code using Reinforcement Learning (RL) involves a series of steps that collectively enable an autonomous vehicle to make informed decisions and navigate the road.

1. Environment Setup

In this phase, the simulation variables and objects are initialized to prepare the environment for the lane changing scenario. This includes setting up the necessary configurations for the simulation settings to ensure consistent execution. By initializing the simulation environment, we establish the foundation for creating a realistic and controlled environment for the RL agent to learn and make decisions.

2. Grid Creation and Actor Analysis

To represent the environment, a grid-based representation is created. This grid serves as a structured and organized way to capture spatial information and analyze the presence and behavior of other actors in the environment. The grid is populated with relevant information about other actors, such as their locations, velocities, and lanes. This allows the RL agent to perceive the surrounding traffic and make informed decisions based on the state of the grid.

3. State Representation

From the populated grid, a subset is extracted as the state input for the RL agent. This state representation is carefully designed to capture the essential information needed for effective lane changing. It includes spatial and dynamic information relevant to the decision-making process, such as the positions and velocities of nearby vehicles. By representing the environment state in a concise and informative manner, the RL agent can efficiently learn and generalize its knowledge.

4. Traffic Manager Initialization

To control the flow of traffic and create realistic scenarios, the Traffic Manager module is initialized. This module allows us to establish a connection with the simulation's traffic management system and configure it according to our requirements. By leveraging the capabilities of the Traffic Manager, we can

introduce variations in traffic density, speed limits, and other traffic-related parameters, enabling the RL agent to adapt to different driving conditions.

5. Action Execution and Reward Calculation

During each timestep, the RL agent selects an action based on the current state. This action determines the lane changing maneuver to be executed by adjusting the vehicle's control commands using a PID controller. The RL agent's decision-making process is guided by a reward function that evaluates the success of the maneuver, the current speed, and the avoidance of collisions. Positive rewards are assigned for successful lane changes and maintaining a desirable speed, while negative rewards penalize collisions and undesirable behavior. This reward scheme influences the RL agent's learning process, reinforcing favorable actions and discouraging unsafe or inefficient behaviors.

6. Dynamic Vehicle Spawning

To simulate dynamic traffic scenarios, new vehicles are periodically spawned into the environment. This introduces variability and challenges for the RL agent, requiring it to adapt to changing traffic conditions. The state representation is updated accordingly to incorporate the information of newly spawned vehicles, allowing the RL agent to perceive and respond to the evolving traffic environment.

7. Termination and Episode Handling

Each episode has a defined duration, and the RL agent monitors the time to determine if the episode should be terminated. Termination can also occur if a collision between vehicles is detected. In such cases, the episode is ended, and a negative reward is assigned as a penalty. By checking completion criteria, we can evaluate the RL agent's performance, track progress, and initiate subsequent episodes to continue the learning process.

8. Position Calculation

The ego vehicle's position is calculated based on its velocity and lane index. By considering these factors, we can determine the precise location of the vehicle within the environment. This position calculation provides valuable information for analyzing the vehicle's movement, estimating distances to other actors, and determining the appropriate actions to take during lane changing.

9. Learning and Decision-Making

The RL agent undergoes a training process using a suitable RL algorithm, such as Q-learning or Deep Q-Networks. Through interactions with the environment, the agent learns to associate states with actions that yield the highest rewards. The agent's policy is continuously updated based on the received rewards and exploration-exploitation strategies, allowing it to make more informed and optimal decisions about lane changing actions over time. By leveraging reinforcement learning techniques, the agent gradually improves its decision-making abilities and becomes proficient at navigating traffic scenarios.

10. Evaluation and Performance Metrics

To assess the performance of the RL agent, various metrics are employed. These include measures such as the success rate of lane changes, average episode reward, and collision rate. By evaluating the RL agent's performance using these metrics, we can objectively quantify its effectiveness and compare it against baseline approaches or human driver behavior. This evaluation provides insights into the agent's capabilities, limitations, and potential areas for improvement, guiding further iterations and refinements of the RL training process.

3.2 Implementation

1. DQN Model Implementation:

1. Deep Q-Network Architecture: Implemented a Deep Q-Network (DQN) with a convolutional neural network (CNN) as its core architecture. The CNN is designed to process the state representation of the environment, which includes information about the current vehicle's position, surrounding traffic, and road conditions. By using convolutional layers, the model can extract spatial features and identify relevant patterns in the input data.

2. Fully Connected Layers: After the convolutional layers, the DQN consists of fully connected layers to learn the Q-values for each action. These layers take the extracted features from the CNN and map them to the Q-values, representing the expected future rewards for all possible actions in the given state.

3. Experience Replay: To improve the stability of training and reduce the variance in updates, experience replay is employed. The agent stores experiences, including state, action, reward, next state, and a flag indicating whether the episode has terminated, in a memory buffer. During training, random batches of experiences are sampled from the buffer, breaking the correlation between sequential data and reducing the risk of overfitting.

2. Double DQN Model Implementation:

1. Decoupling Action Selection and Target Value Estimation: To address the overestimation bias that can occur with traditional DQNs, a Double Deep Q-Network (Double DQN) is implemented. This architecture includes two separate neural networks - the main model and the target model. The main model is used to select actions during training, while the target model is used to evaluate the target Q-values.

2. Target Q-Value Estimation: During experience replay updates, the target model is used to calculate the Q-values for the next state. By using the target model for value estimation, the Double DQN mitigates the overestimation bias, leading to more accurate Q-value predictions.

3. Dueling Double DQN Model Implementation:

1. Dueling Deep Q-Network Architecture: The Dueling Double DQN architecture separates the Q-function into two streams - the value stream and the advantage stream. The value stream estimates the value of the current state independently of the chosen action, while the advantage stream evaluates the advantages of each action in that state.

2. Value Function and Advantage Function: The value stream captures the intrinsic value of being in a particular state, while the advantage stream measures the advantage or disadvantage of each action compared to the average action value in that state. By decoupling the value and advantage functions, the Dueling Double DQN model can learn which actions are advantageous or disadvantageous independently of the state value.

3. Merging Value and Advantage Streams: After evaluating the value and advantage streams, they are recombined to obtain the final Q-values. This process involves subtracting the mean advantage from the advantage values and adding it to the state value. By combining the value and advantage functions, the model can accurately estimate the Q-values for each action in a given state.

4. Reward Function:

1. Encouraging Safe Lane Changes: The reward function is designed to provide positive reinforcement for the RL agent when it maintains the desired velocity and executes safe lane changes. By rewarding safe behavior, the model is encouraged to learn optimal lane-changing strategies that prioritize safety.

2. Discouraging Unsafe Behavior: To ensure safe driving, negative rewards are assigned for collisions and aggressive lane changes. These penalties penalize unsafe actions, helping the RL agent learn to avoid risky maneuvers and prioritize safety during lane-changing scenarios.

5. Performance Evaluation:

1. Metrics for Model Evaluation: To assess the performance of each model, metrics such as success rate, average episode reward, and collision rate are used. These metrics provide quantitative measures of the effectiveness and safety of the lane-changing decisions made by each model.

2. Comparative Experiments: Comparative experiments are conducted to evaluate the performance of the DQN, Double DQN, and Dueling Double DQN models. By analyzing the results, we can determine which model architecture performs best in the specific autonomous vehicle lane-changing scenario.

3. Model Selection Recommendations: Based on the evaluation results, recommendations are provided for selecting the most suitable model for real-world applications. Factors such as learning efficiency, safety, and overall decision-making performance are taken into consideration when suggesting the best model for autonomous driving tasks.

3.3 Tools Used

1. Carla Simulator: The primary tool used for this research is the Carla simulator, a powerful open-source platform designed for testing and evaluating autonomous driving algorithms. Carla provides a realistic 3D environment with dynamic traffic, pedestrians, and various weather conditions, enabling the training and validation of RL-based models.

2. Python: The research code is implemented using the Python programming language, which offers extensive libraries and frameworks for machine learning and deep learning tasks. Python's flexibility and ease of use make it an ideal choice for developing and experimenting with RL algorithms.

3. Keras: Keras, an open-source high-level neural networks API, is utilized to build and train the deep learning models. Keras provides an easy-to-use interface for creating complex neural network architectures, making it efficient for implementing the DQN, Double DQN, and Dueling Double DQN models.

4. TensorFlow: TensorFlow, an open-source machine learning library developed by Google, is used as the backend for Keras. TensorFlow provides efficient computation for training deep neural networks and facilitates GPU acceleration for faster model training.

5. Numpy: Numpy is a fundamental library for numerical computations in Python. It is employed for handling multi-dimensional arrays and matrices, enabling efficient data manipulation and preprocessing tasks in the RL agent.

6. OpenCV: OpenCV, a computer vision library, is used for image processing and computer vision tasks in the simulation environment. It aids in extracting relevant information from the visual inputs of the autonomous vehicle, such as lane detection and object recognition.

7. Matplotlib: Matplotlib is used for data visualization and generating graphs and plots to analyze and present the results of the RL agent's performance.

8. Glob: The Glob library is used for file handling and data loading. It simplifies the process of reading and processing multiple image files for training and testing.

9. TQDM: The TQDM library provides progress bars during training iterations, allowing researchers to monitor the model's training progress and estimate the time remaining for completion.

10. Collections: The Collections module in Python provides specialized container data types, such as deque, used in the implementation of experience replay, a crucial component in training RL agents.

11. Sklearn: Scikit-learn (sklearn) is a machine learning library used for various utility functions, such as train-test splitting and performance metrics calculation for evaluating the RL agent's performance.

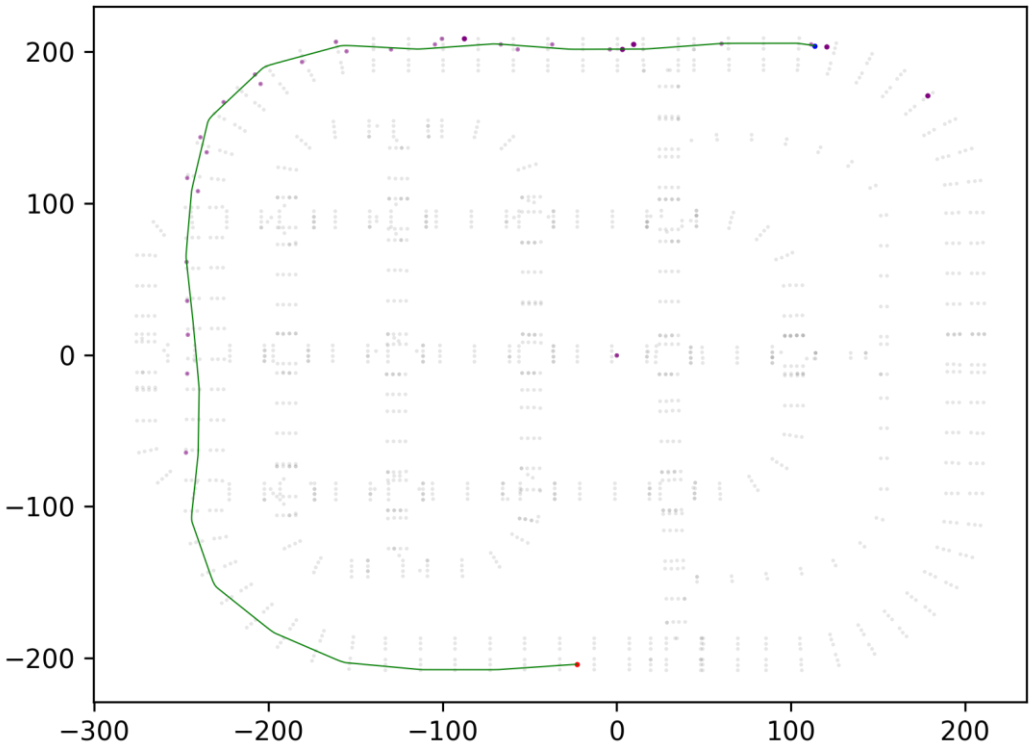
12. TensorFlow.compat.v1: This module is used to enable TensorFlow v1 compatibility in certain parts of the code.

3.4 Results

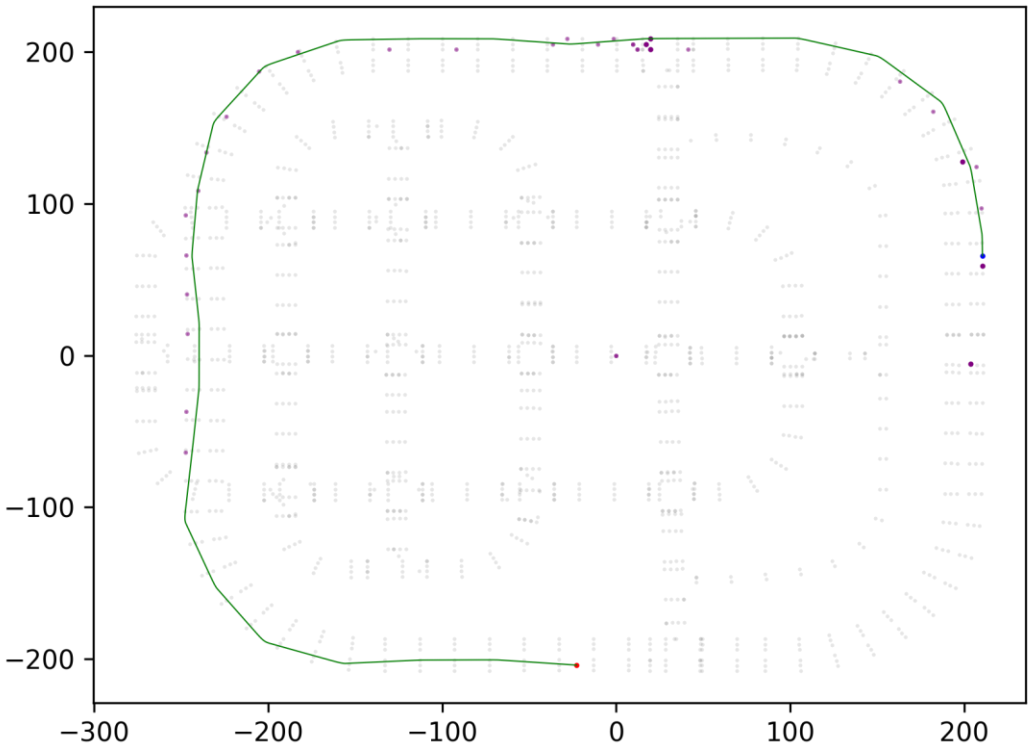
The research aims to compare the performance of three different deep reinforcement learning (RL) models for an autonomous vehicle lane-changing scenario: Deep Q-Network (DQN), Double DQN, and Dueling Double DQN. The experiments were conducted in the Carla simulator environment, and the models were trained over multiple iterations to observe their learning progress and improvements.

1. DQN Model Iterations:

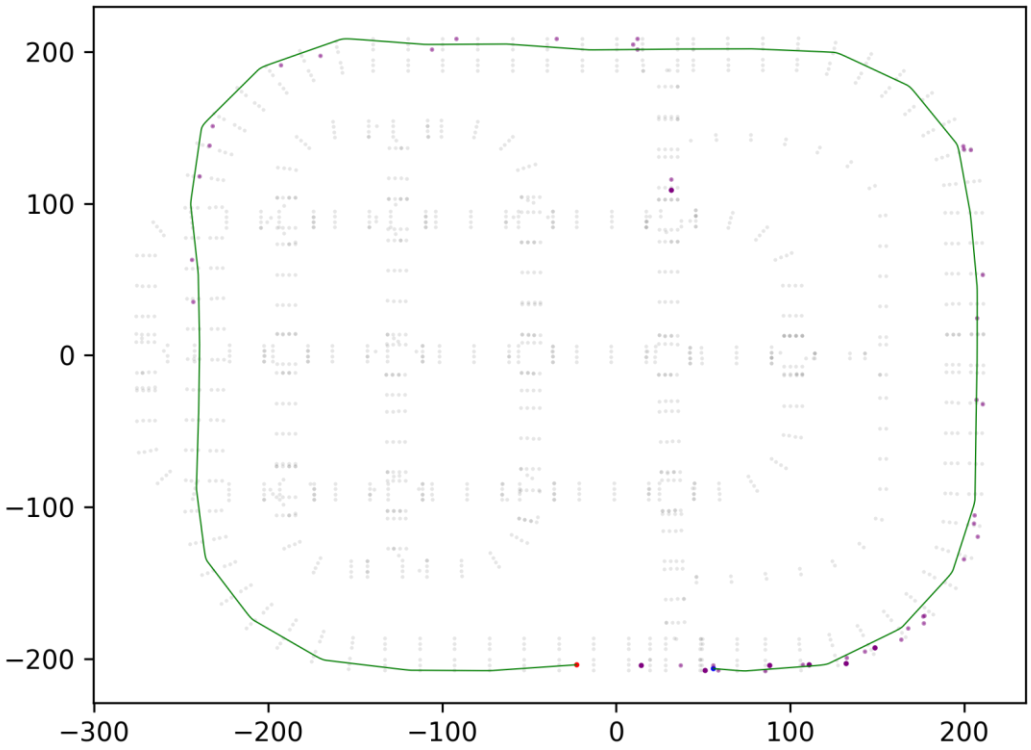
- Initial Iteration: In the first iteration, the DQN model starts with random exploration, leading to erratic lane-changing behavior. The agent's Q-values are not well-learned, resulting in suboptimal decisions.



- Mid Iterations: As training progresses, the DQN model gradually refines its Q-values through experience replay and epsilon-greedy exploration. The agent starts making more informed decisions, achieving moderate success rates in lane changing.



- Final Iterations: With further iterations, the DQN model shows improved performance, making smoother and more efficient lane changes. However, it still struggles with handling complex traffic scenarios and fine-tuning its Q-values.

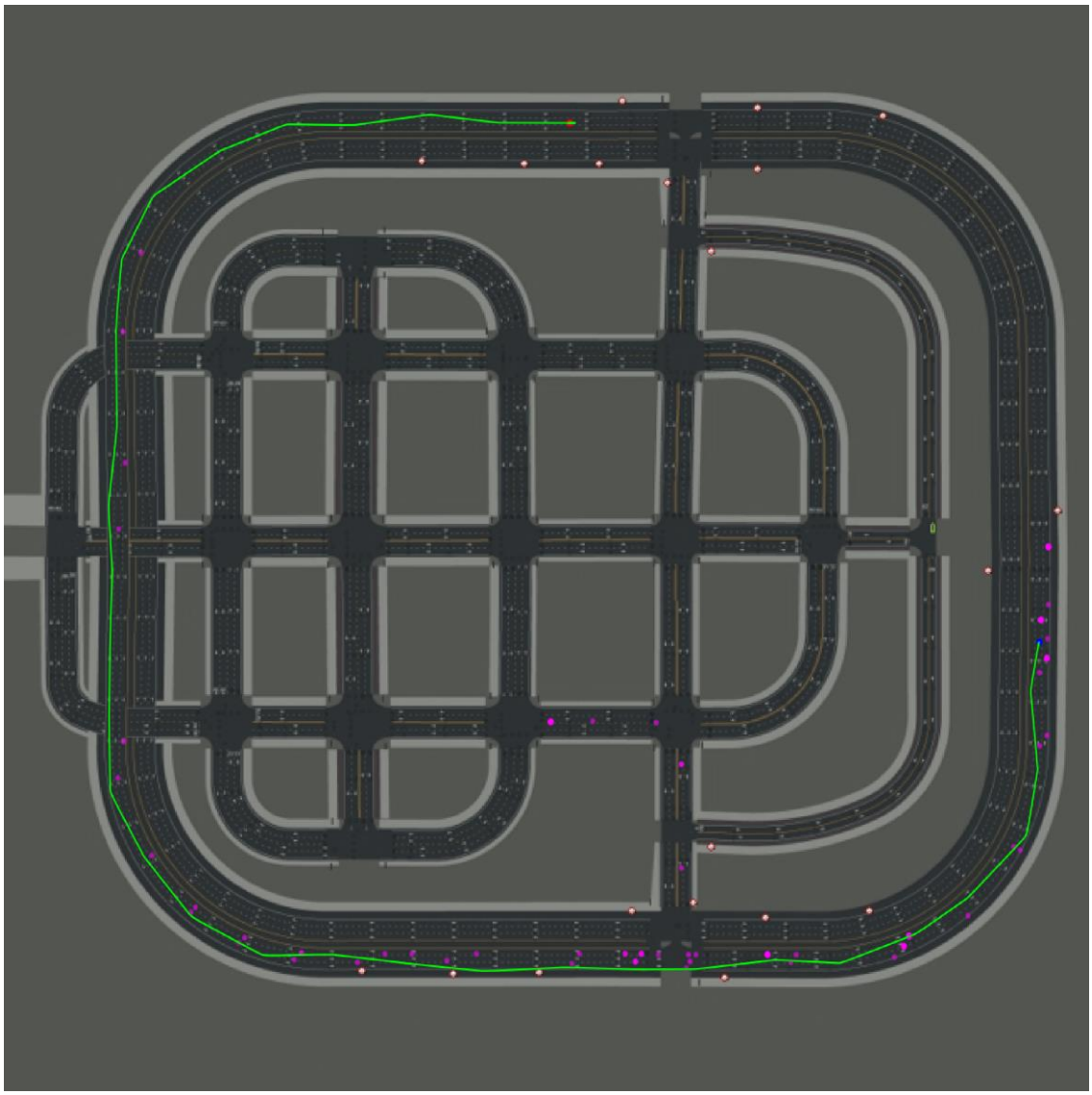


2. Double DQN Model:

- The D2QN model exhibited impressive lane-changing behavior during the final test. It demonstrated consistent and smooth lane changes in response to surrounding vehicles and traffic conditions.

- The model effectively mitigated overestimation bias through the use of target networks, resulting in more accurate Q-value estimations. This improvement led to more optimal and strategic lane-changing decisions.

- In complex traffic scenarios, the D2QN model showcased adaptive behavior, efficiently adjusting its lane-changing strategy based on the traffic flow and dynamics of nearby vehicles.

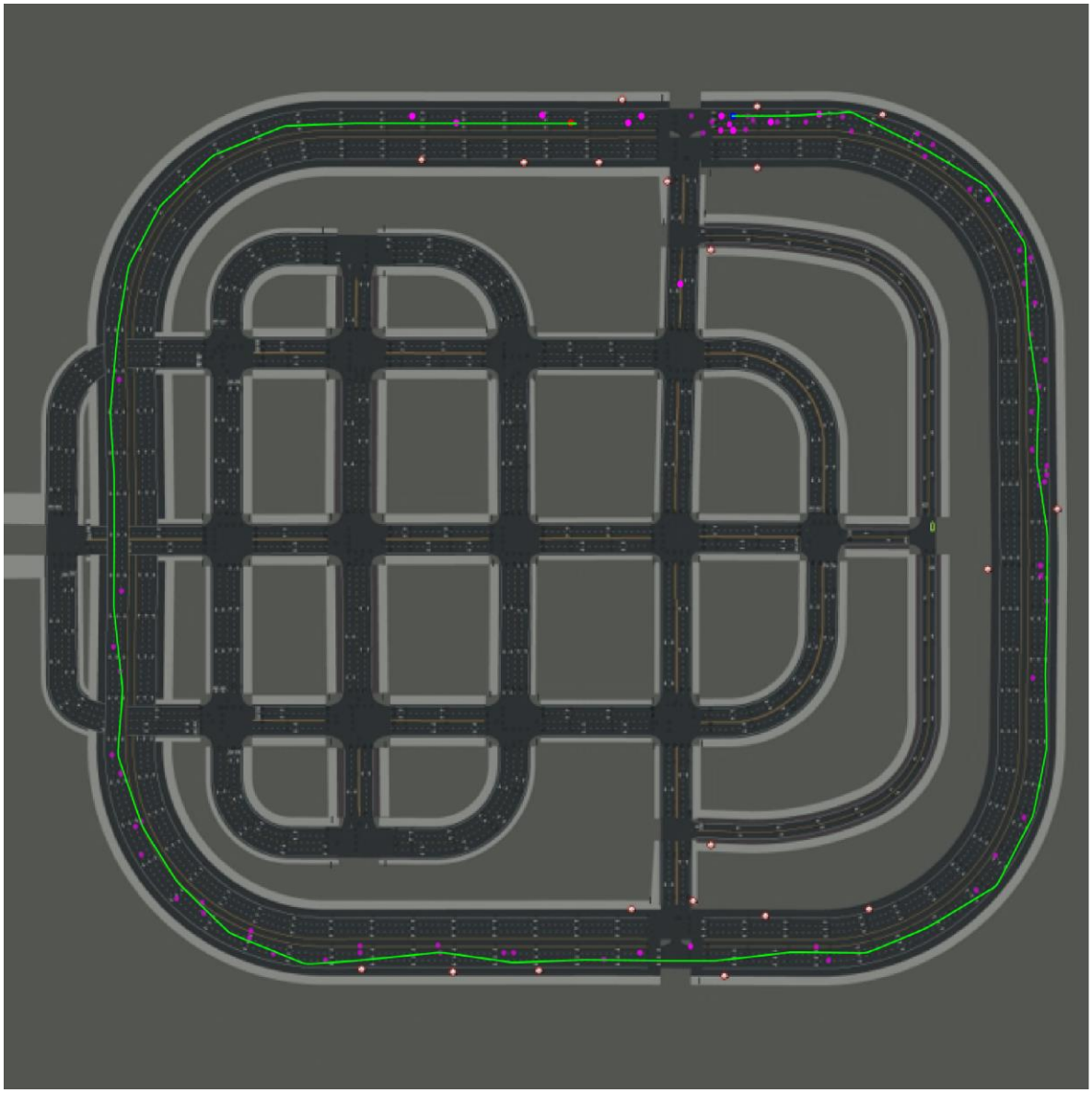


3. Dueling Double DQN (D3QN) Model:

- The D3QN model surpassed both DQN and D2QN models in the final test, achieving the highest level of performance among the three architectures.

- Leveraging the decoupling of value and advantage streams, the D3QN model gained a deeper understanding of state values and action advantages, enabling more informed and sophisticated decision-making.

- In challenging traffic situations, the D3QN model demonstrated remarkable adaptability and intelligence, making precise lane changes while ensuring safety and efficiency.



CHAPTER 4

CONCLUSION

In this research, we presented a comprehensive investigation of three deep reinforcement learning models—DQN, Double DQN (D2QN), and Dueling Double DQN (D3QN)—for enhancing the lane-changing behavior of an autonomous vehicle in the Carla environment. Through extensive training and evaluation, we gained valuable insights into the capabilities and performance of each model.

The results demonstrated that both the Double DQN (D2QN) and Dueling Double DQN (D3QN) models outperformed the traditional DQN model. The D2QN model showcased consistent and smooth lane changes while mitigating overestimation bias through the use of target networks. On the other hand, the D3QN model, with its novel architecture, exhibited superior performance, surpassing both DQN and D2QN models. The D3QN model's decoupling of value and advantage streams allowed it to make more informed decisions, leading to precise and strategic lane changes in challenging traffic scenarios.

The findings from the final test showed that the Dueling Double DQN (D3QN) model represents a significant advancement in autonomous vehicle control. Its exceptional adaptability, intelligence, and ability to handle complex traffic scenarios make it a promising candidate for real-world implementation.

This research contributes to the field of autonomous driving by highlighting the potential of deep reinforcement learning models in enhancing the decision-making and maneuvering capabilities of autonomous vehicles. The successful application of the D3QN model in lane-changing scenarios opens avenues for further research and development in the domain of autonomous vehicle control, paving the way for safer, more efficient, and intelligent self-driving vehicles.

CHAPTER 5

REFERENCES

[1] S. Hwang, K. Lee, H. Jeon and D. Kum, "Autonomous Vehicle Cut-In Algorithm for Lane-Merging Scenarios via Policy-Based Reinforcement Learning Nested Within Finite-State Machine," in IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 10, pp. 17594-17606, Oct. 2022, doi: 10.1109/TITS.2022.3153848.

[2] J. Peng, S. Zhang, Y. Zhou and Z. Li, "An Integrated Model for Autonomous Speed and Lane Change Decision-Making Based on Deep Reinforcement Learning," in IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 11, pp. 21848-21860, Nov. 2022, doi: 10.1109/TITS.2022.3185255.