UNIVERSITY OF LIVERPOOL

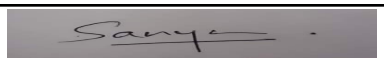
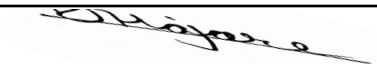
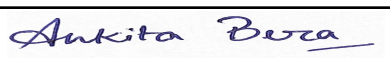
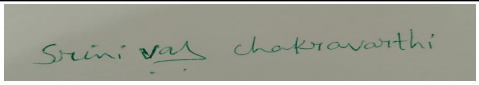
REINFORCEMENT LEARNING FOR AUTONOMOUS VEHICLE

COMP530 - MSC GROUP PROJECT

PORTFOLIO (PROJECT COMPLETION REPORT)

SUPERVISOR: ANTONIA TSILI (a.tsili@liverpool.ac.uk)

Team Members:

Debajyoti Ghosh	201626481	
Sanya Upadhyay	201680389	
Digvijay Ashok Hajare	201684761	
Vijay Kunchala	201674693	
Rahul Arun Nawale	201669264	
Srinivasan Rajagopal	201666283	
Ankita Bera	201671685	
Subrahmanya Srinivasa Chakravarthi Sabbarapu	201644447	

Introduction

Imagine you're embarking on an adventurous road trip. Your destination is clear - a place where autonomous vehicles can effortlessly navigate around obstacles. The question is, which route will lead you there most effectively?

In this journey of ours, the world of technology was our landscape, and our vehicle was an innovative project, fueled by the power of deep reinforcement learning. The destination? To demonstrate the effectiveness of a chosen reinforcement learning algorithm in mastering the obstacle avoidance task within an autonomous vehicle.

Although our project's initial aim encompassed both navigation control and obstacle avoidance, the limited resources and time constraints we encountered led us to make a focused decision to prioritize obstacle avoidance. This strategic shift allowed us to channel our efforts effectively and maximize the potential outcomes of our project within the given resources and timeframe.

Aim:

The **aim** of our project is to explore reinforcement learning algorithms and utilize Deep Q-Networks (DQN) to guide an autonomous vehicle through a simulated environment while avoiding obstacles.

Objectives :

To reach this ambitious destination, we charted out clear objectives. Just like any explorer worth their salt, we knew that a successful journey needed a well-thought-out plan.

1. **Exploring the Landscape of Reinforcement Learning Algorithms**: Before we started driving, we needed to understand the terrain. So, we began with a comprehensive study of various deep reinforcement learning algorithms. We compared their strengths, weaknesses, and applicability to our task.
2. **Choosing the Best Route - DQN**: After a thorough comparison, we found our optimal path - the Deep Q-Network. We chose DQN for its ability to handle high-dimensional state spaces, its stability during learning, and its proven track record in tasks similar to ours.
3. **Setting up the Simulation Environment**: Next, we needed a testing ground, so we set up a dynamic simulation environment filled with obstacles in Webots.
4. **Designing and Implementing the DQN Model**: With the terrain and the route set, we moved on to designing and implementing our DQN model. We defined the state space, action space and reward function, fine-tuned the model, and prepared it for the journey.
5. **Training and Evaluating the Model**: The final leg of our journey was to train the model, observe its performance, and evaluate its effectiveness in obstacle avoidance.

Milestones :

To ensure we didn't lose our way, we set up milestones - landmarks to guide us and measure our progress.

1. **Deep Reinforcement Learning Research Completion (Week 1-2)**: Our first objective is to conduct a comprehensive study of various deep reinforcement learning algorithms and compare their strengths, weaknesses, and applicability to our task. By the end of week 2, we aim to have completed this research.

2. **Selection of DQN (Week 3)**: After a thorough comparison, we aim to select the Deep Q-Network as our algorithm of choice by the end of week 3.
3. **Simulation Setup (Week 4-6)**: Our next objective is to set up a dynamic simulation environment filled with obstacles in Webots. We aim to start this work by week 4 and have the environment fully set up by the end of week 6.
4. **Model Design Start (Week 6-7)**: This week we hoped to start off our work on the DQN model.
5. **Model Design Completion (Week 7-9)**: By this week, we hoped to have a fully defined model, complete with state space and reward function.
6. **Model Training Start (Week 9-10)**: We aim to commence model training by week 10.
7. **Model Training Completion (Week 10-12)**: By week 12, we aimed to have a fully trained model ready for testing.
8. **Model Evaluation and Result Analysis (Week 12-14)**: The final milestone was the evaluation and analysis of our model's performance.

GANTT CHART

	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1		Exploring the Landscape of Reinforcement Learning Algorithms	20 days	Mon 27/02/23	Fri 24/03/23		Rahul[88%],Ankita[88%], Digvijay[88%], Srinivasan[88%], Sanya[88%],Debajyoti[88%],Srini
2		Choosing the Best Route - DQN	4 days	Sat 25/03/23	Wed 29/03/23	1	Debajyoti
3		Deep Reinforcement Learning Research Completion	1 day	Sat 25/03/23	Sat 25/03/23	1	
4		Selection of DQN	3 days	Mon 27/03/2	Wed 29/03/2	3	
5		Setting up the Simulation Environment	29 days	Mon 27/02/23	Wed 05/04/23		Digvijay,Rahul
6		Simulation Setup Start	0 days	Mon 27/02/23	Mon 27/02/23		
7		Simulation Setup Completion	0 days	Wed 05/04/23	Wed 05/04/23	6	
8		Designing and Implementing the DQN Model	14 days	Wed 05/04/23	Sun 23/04/23	4,7	Rahul
9		Model Design Start	0 days	Wed 05/04/2	Wed 05/04/2	4,7	
10		Model Design Completion	0 days	Sun 23/04/23	Sun 23/04/23	9	
11		Training and Evaluating the Model	14 days	Tue 25/04/23	Fri 12/05/23	10	Rahul,Ankita,Srinivas,Vijay, Srinivasan
12		Model Training Start	0 days	Tue 25/04/2	Tue 25/04/2	10	
13		Model Training Completion	0 days	Fri 12/05/23	Fri 12/05/23	12	
14		Model Evaluation and Result Analysis	0 days	Fri 12/05/23	Fri 12/05/23	13	Debajyoti,Digvijay,Rahul,Sanya

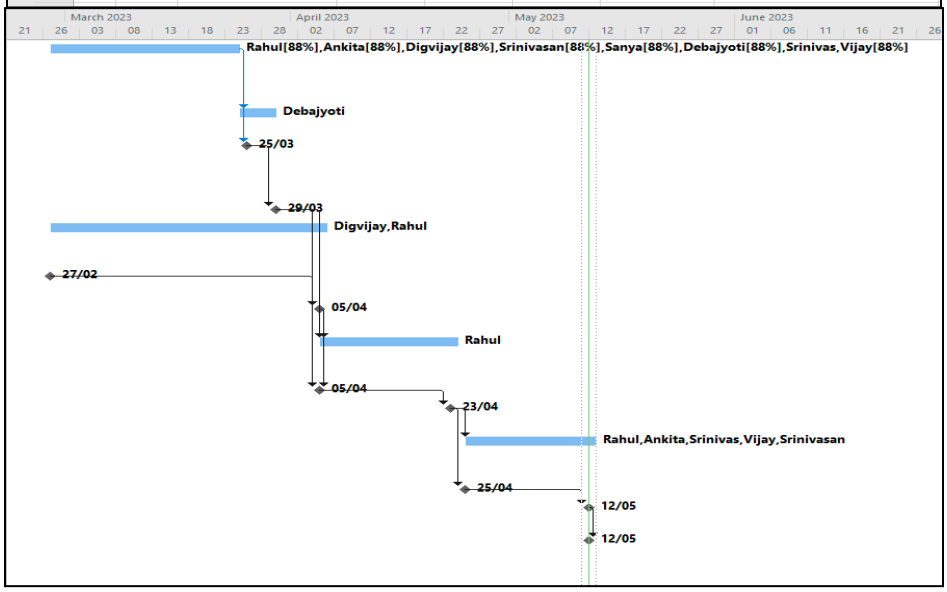


CHART 1: GANTT CHART(OBJECTIVES AND MILESTONES)

To ensure we stay on track, we have set up clear and specific milestones to guide us and measure our progress. In case of any unexpected setbacks or delays, we will also have contingency plans in place, such as, If there are delays in the project timeline due to unforeseen circumstances such as technical issues, team member unavailability, or other reasons, we could adjust the project timeline accordingly or assign additional resources to complete the work.
Additionally, we will measure progress along the way and define success criteria for each milestone to ensure we are making steady progress towards our ultimate destination.

Details of the team members and a summary of their roles

Data Collection and initial Research:

- **Members**: Rahul, Ankita, Srinivas, Vijay, Digvijay, Srinivasan, Sanya, Debajyoti.
- **Task**: Every member carried out research about the project and was responsible for bringing up individual ideas and concerns as the project progressed.

Deep Q-Learning Algorithm Implementation

- **Members**: Debajyoti, Rahul
- **Task**: Responsible for implementing the deep Q-learning algorithm, including selecting appropriate hyperparameters and optimizing the algorithm for the specific task of autonomous driving.

Autonomous Vehicle Simulation:

- **Members**: Digvijay, Rahul
- **Task**: Building a simulation environment for the autonomous vehicle to test the deep Q-learning algorithm.

Evaluation and testing :

- **Members**: Rahul, Ankita, Srinivas, Vijay, Srinivasan
- **Task**: Responsible to test correctness of algorithm implementation and evaluate as well summarize performance of RL model.

Project Management and Coordination:

- **Members**: Sanya
- **Tasks**: Responsible for managing the overall project, including setting deadlines, arranging meetings and ensuring that each member of the group is on track.

Project Record:

- **Members**: Srinivasan, Sanya
- **Tasks**: Responsible for creating project reports, presentation slides and keeping meeting records.

Presentation:

- **Members**: Vijay, Debajyoti, Rahul
- **Tasks**: Responsible for presenting project ideas, implementation and results using slideshow or videos.

Preparation of project portfolio:

- **Members**: Debajyoti, Rahul, Sanya, Digvijay
- **Tasks**: Preparing the final portfolio for the project.

An Overview of the Outcome of the Project

We set out to develop an autonomous vehicle capable of avoiding obstacles in a dynamic environment using a Deep Q-Network (DQN). Today, we look upon our creation with pride, acknowledging the transformation it has undergone. The once novice driver now skillfully maneuvers around obstacles, demonstrating the power of deep reinforcement learning and the effectiveness of DQN.

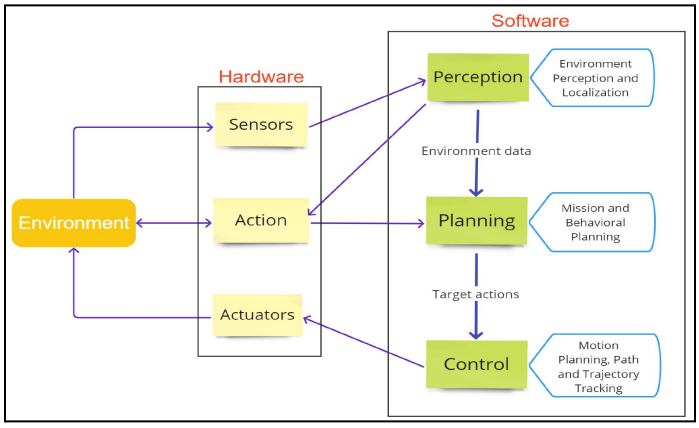


FIG. 1: ARCHITECTURE OF AUTONOMOUS VEHICLE SYSTEM

Our autonomous vehicle, with its newfound proficiency, is designed for anyone who wishes to venture into the cutting-edge world of autonomous technology. It could be a research institution investigating reinforcement learning's potential in real-world scenarios or a tech company exploring autonomous vehicles' capabilities. Our vehicle is not just a demonstration of DQN's power; it's a platform for discovery, innovation, and progress.

Those who choose to utilize our autonomous vehicle will find it offers more than just a tool for obstacle avoidance. It provides a window into the world of reinforcement learning, showcasing how DQN can be trained to handle complex tasks. It offers a foundation on which further advancements in autonomous technology can be built.

In essence, the vehicle provides a compelling answer to the question, "why use it?" It offers a practical application of reinforcement learning, demonstrates the possibilities of DQN, and serves as a stepping stone towards more advanced autonomous vehicle technologies. The vehicle we have created through this journey is not just an end-product; it's a beacon lighting the way for future research and applications in the field of autonomous vehicles.

Implementation & Challenges of the Project

We detail various parts of our implementation including simulation environment elements and challenges faced throughout the project in the following section.

Selection of algorithm:

At the start of our journey, we faced a crucial decision in choosing the best deep reinforcement learning algorithm to guide us. We researched extensively, diving deep into the world of reinforcement learning, examining each algorithm's strengths and weaknesses. Our search eventually led us to two promising candidates: Proximal Policy Optimization (PPO) and Deep Q-Networks (DQN).

Algorithm	Pros	Cons	State Space	Action Space	Training Time	Generalization
Q-learning	Simple and easy to implement	Slow to converge, may not scale well to large state spaces	Discrete	Discrete	Medium	Low
Deep Q-learning	Handles more complex state spaces, faster convergence	Can be unstable, prone to overestimation of action values	Continuous	Discrete/Continuous	Long	Medium
Policy gradients	More stable and efficient learning	High variance, slow to converge	Continuous	Continuous	Long	High
Actor-critic methods	Faster convergence, more stable learning	More complex to implement	Continuous	Continuous	Long	Medium
Model-based RL	More efficient learning, better generalization	Computationally expensive, may not scale well to large state spaces	Continuous	Continuous	Long	High

TABLE 1 : COMPARISON BETWEEN DIFFERENT ALGORITHMS

In Table 1, we compared the features of different algorithms and found that PPO offered a stable and efficient policy-based approach, while DQN excelled in handling high-dimensional state spaces and had a proven track record in similar tasks. In the context of reinforcement learning algorithms, stable typically refers to the algorithm's ability to converge to an optimal solution reliably and efficiently without becoming unstable or oscillating between suboptimal solutions. In other words, a stable algorithm should be able to learn from its experience and improve its performance over time in a smooth and predictable way.

We deliberated on various factors such as ease of implementation, adaptability to our specific problem, and the potential challenges each algorithm posed. Ultimately, we chose DQN as our guide for this journey, drawn to its compatibility with our task and the intriguing challenge it presented.(Sutton & Barto, 2018)

DQN is a reinforcement learning algorithm that builds upon Q-Learning, utilizing deep learning techniques to navigate high-dimensional state spaces. It can be thought of as a skilled off-road driver who can navigate complex terrain where traditional Q-Learning struggles. This ability is due to the use of neural networks, which allow DQN to approximate the Q-value function($Q(s, a) = r + \gamma * \max[Q(s', a')]$), a measure of the expected reward for an action in a given state. It's similar to predicting the best path to take at any given point in our journey (Sutton & Barto, 2018)

Implementing DQN felt like building our vehicle from scratch, each piece carefully crafted and assembled.

Hyperparameters :

Hyperparameters	Values
Gamma	0.99
Epsilon	1.0
Epsilon_min	0.01
Epsilon_decay	0.995
Learning_rate	0.001
Batch_size	32
Memory_size	100000

Deep Q Model Information :

Model Architecture	3 Fully Connected Layers
Hidden Layer Activation	ReLU
Optimizer	Adam
Learning Rate	0.001
Input Size	Size of State Space (14)
Output Size	Size of Action Space (6)

Other Information :

Input to Neural Network	State represented as 1D array
Output of Neural Network	Q-value for each action in action space
Exploration Rate	Starts high (1.0) and decays over time
Update Frequency	Target network updated every 100 episodes

Layers and Optimizers Used :

Fully Connected Layers	24-48-24
Activation Function	ReLU
Optimizer	Adam (Learning Rate: 0.001)

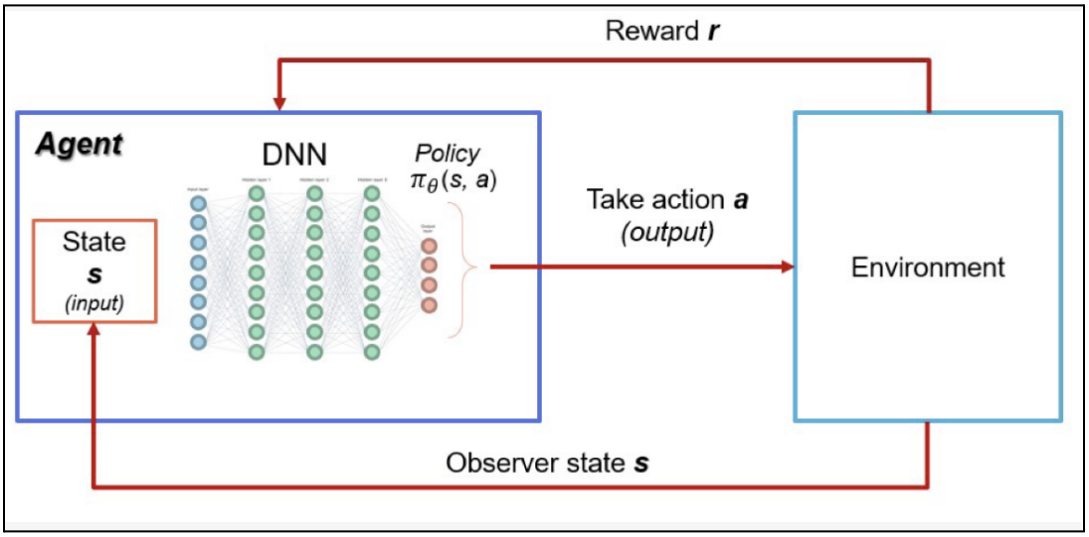


FIG. 2: DQN LEARNING PROCESS

(MDPI. (2022). Obstacle Avoidance Design Architecture [Digital image]. Retrieved from https://www.mdpi.com/wevj/wevj-13-00239/article_deploy/html/images/wevj-13-00239-g005-550.jpg)

Here's the table representing the rewards for different scenarios in the reinforcement learning environment:

Scenario	Reward
Collision	-50
Staying on the road without collision	+0.1
Previous Lane != Current Lane	-0.01
Close to other vehicle (obstacle) without crashing	-0.01
Completed 1 mile without crashing	+100

The reward system in this project comprises three types of rewards: the primary reward (Rm), penalty reward (Rp), and additional reward (Rex). Each of these rewards serves a specific purpose during the training phase. The overall reward, denoted as R_T, is calculated by combining these individual rewards: $R_T = R_m + R_p + R_{ex}$ (Terapaptommakol et al., 2022).

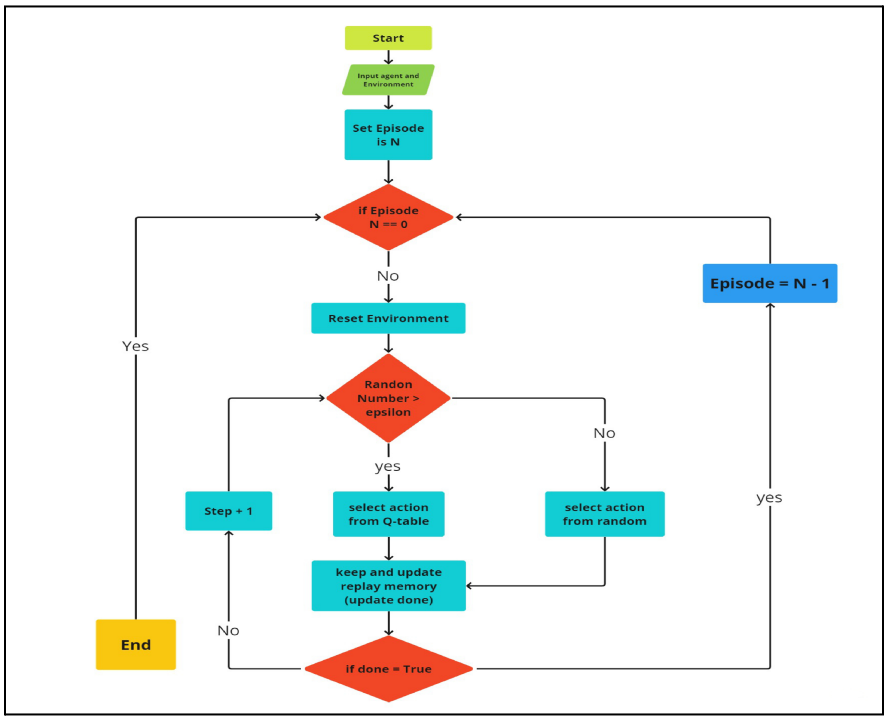


FIG. 3: PROCESS TO SELECT THE ACTION IN DQN MODEL

The creation of our simulation environment was much like setting the stage for a grand performance. **Webots**, our chosen platform, provided us with an ideal playground for our autonomous vehicle training. Our simulation environment was carefully crafted to reflect a realistic road scenario. Here's a concise description of the simulation environment:-

- **Road Configuration:** Our simulated environment consisted of a straight road with three lanes, mimicking real-world highway conditions.
- **Agent/Driver:** The central character of our training process was a four-wheeled car acting as the agent/driver. This car would be the focus of our reinforcement learning efforts.

- **LIDAR Sensors:** To enable the agent to perceive its surroundings, we equipped the car with 12 LIDAR sensors. These sensors had a maximum range of 20 meters, allowing the agent to detect obstacles in its vicinity accurately.
- **Obstacles:** Other vehicles served as obstacles in our simulation, mimicking real traffic scenarios. These vehicles moved in the same direction as the agent/driver, presenting challenges for it to navigate through.
- **Road Dividers:** Dividers on each side of the road helped maintain lane integrity and provided spatial cues for the agent/driver.
- **Speed Limit:** To replicate highway speeds, we set the maximum speed of the agent/driver to 80 mph, challenging its ability to navigate swiftly.
- **Road Length:** The simulated road extended over a length of 200 miles, offering a long stretch for our agent/driver to navigate and learn from.

With this carefully designed simulation environment, our agent/driver was exposed to a variety of realistic scenarios, enabling it to learn and adapt its behavior for effective obstacle avoidance. To achieve this, we equipped the vehicle with a set of actions and an array of sensors strategically positioned on the vehicle.

The vehicle had a total of **6 actions** at its disposal, allowing it to navigate and adjust its speed according to the situation:

Go to left lane	Go to middle lane	Go to right lane
Speed = 5 mph	Speed = 10 mph	Speed = 20 mph
Speed = 30 mph	Default = 40 mph	

To perceive its surroundings, the vehicle was equipped with **12 sensors** placed at key positions:

Front	Front right 0	Front right 1	Front right 2
Left	Front left 0	Front left 1	Front left 2
right	Rear left	Right right	rear

These sensors provided vital information about the environment, allowing the agent/driver to make informed decisions based on its surroundings.

To detect potential collisions, the vehicle employed a collision detection mechanism. If the value returned by any sensor was less than 15% of its maximum value, it would indicate a potential collision:

- **If sensors["SensorName"].getValue() < 0.15 * sensors["SensorName"].getMaxValue()**

In terms of state representation, the vehicle's perception incorporated information from the sensor array, along with additional state variables. The state space encompassed **14 dimensions**:

Front Sensor	Front Right 0 Sensor	Front Right 1 Sensor
Front Right 2 Sensor	Front Left 0 Sensor	Front Left 1 Sensor
Front Left 2 Sensor	Rear Sensor	Rear Left Sensor
Rear Right Sensor	Right Sensor	Left Sensor
Car Lane Position [0, 1, 2]	Car Speed	

By integrating this comprehensive sensory input, the agent/driver could effectively perceive its surroundings and learn to maneuver in the simulated environment, acquiring the skills needed for successful obstacle avoidance.

Training the model was like embarking on a voyage, guided by the reward function and state space. The state space represented the vehicle's surroundings, sensed by high-dimensional readings from its sensors. The reward function encouraged the vehicle to avoid obstacles and explore its environment, with positive rewards for avoiding collisions, reaching the destination, and negative penalties for collisions.

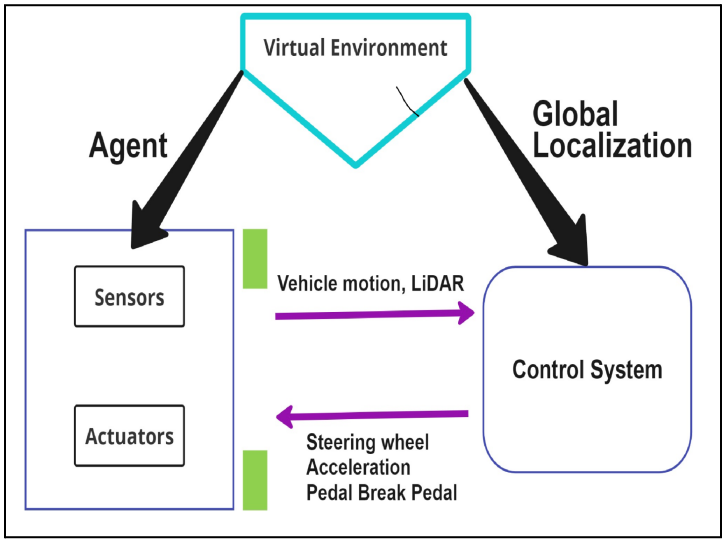


FIG. 4: WEBOTS ARCHITECTURE FOR AUTONOMOUS VEHICLE SIMULATION

The iterative process of training the DQN model involved exploring and exploiting actions, with the neural network's weights updated to improve its ability to predict the best action given a state. As the vehicle learned from its missteps, it relied more on its knowledge to avoid obstacles.

Challenges and Limitations:

Given below are the challenges and how we managed to overcome those -

Data Collection:

- Difficulty in collecting enough data to train the model due to the complexity of the environment and the limited time and resources available.
- The team had to rely on simulations to generate data, which might not be representative of real-world scenarios.

Hyperparameter Tuning:

- Difficulties in finding the optimal values for the hyperparameters, such as learning rate, discount factor, exploration rate, and neural network architecture.
- The team had to experiment with different combinations of hyperparameters, which was time-consuming and computationally expensive.

Testing:

- Ensuring that the model could perform well in various environments was challenging.
- The team had to test the model in different scenarios, such as varying obstacle densities, speeds, and shapes, to evaluate its robustness.

Reward Function:

- Defining a reward function that incentivizes the model to avoid obstacles while reaching the goal was not straightforward.
- The team had to experiment with different reward functions and adjust them based on the model's performance.

Integration:

- Integrating the code with the simulation environment was challenging
- The team had to import environment-related variables and ensure compatibility between the simulator and the model.

Deep Q Model:

- Building a deep Q model for more than six actions and more than 15 states with two actions to take at a time was challenging.
- The team had to design a neural network architecture that could handle the high-dimensional input space and the large number of possible actions.

Lack of Prior Work:

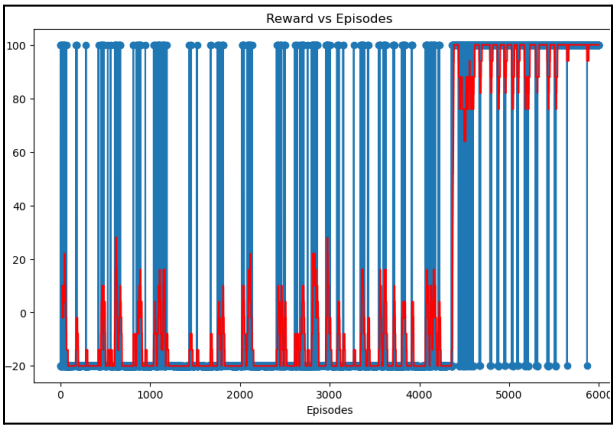
- Not much work has been done in similar areas, making it difficult to refer and learn from past papers and projects.
- The team had to rely on trial and error and intuition to overcome some of the challenges.

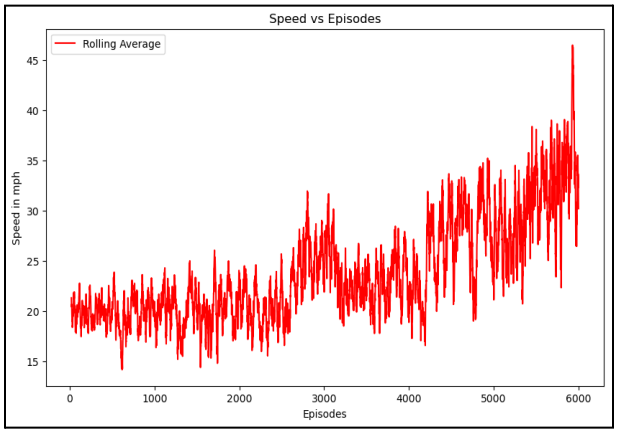
Hardware Requirements:

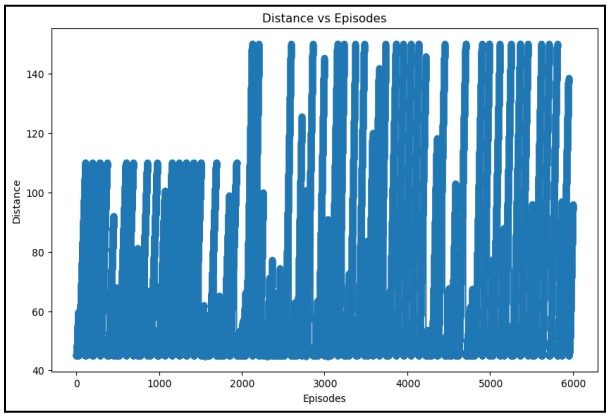
- The simulation environment required Ubuntu with GPU, which none of the team members had.
- The team had to either use cloud computing or acquire the necessary hardware, which added to the project's cost and complexity

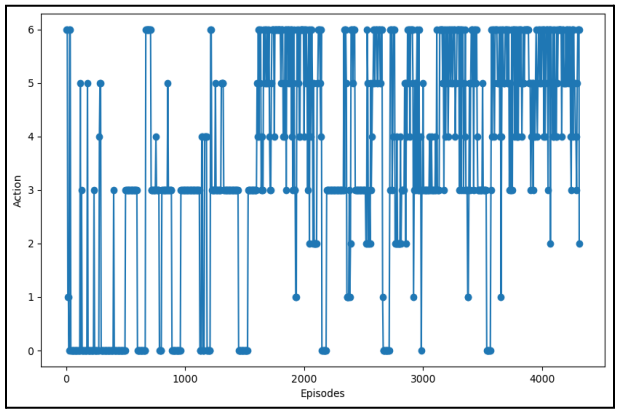
Data, analysis & interpretation

The data gathered during the project included several key metrics: reward over episodes, number of obstacles avoided, and vehicle performance in unfamiliar environments. During the training phase, the reward over time was tracked, serving as an indicator of the vehicle's performance.

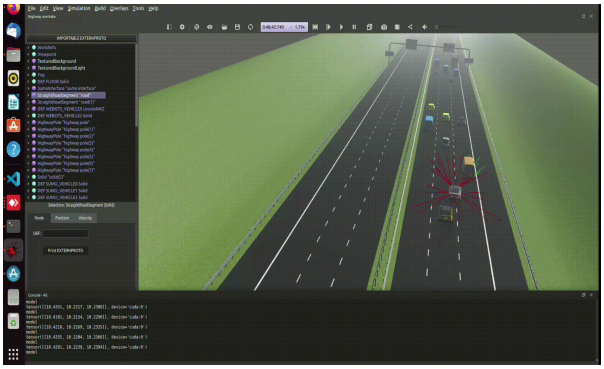








In the first 2000 episodes, rewards were quite low due to the learning curve and penalties associated with crashes. As the model continued to learn and improve, it gradually began to receive positive rewards for maintaining its course without collisions. However, after 6000 episodes, the vehicle encountered difficulty navigating without crashing and only reached the end of the map a handful of times, signaling the need for modification of the model's hyperparameters and techniques.



GIF 1 : DURING TRAINING

GIF 2 : TRAINED MODEL

https://drive.google.com/drive/folders/17D1D5EzYAYjZgUjB_UQKy-HPSo6sBSTz?usp=share_link

The number of obstacles avoided was a key metric during training. The vehicle's speed over time also showed a notable pattern. Initially, the vehicle maintained a slower speed, but as it learned and adapted, it increased its pace. This suggests potential for improved speed with further adjustments to the model.
Distance over time was another metric of interest. Initially, the vehicle covered minimal distance, with a maximum distance of 115 set for the first 2000 episodes to encourage exploration of actions that maximize rewards. After the vehicle learned to achieve this distance, the maximum was increased to 145. Although the vehicle was able to reach the end of the course a few times after 6000 episodes, the learning curve was slower than desired.

Actions taken per episode were also tracked, with actions labeled as: 0 for "go to left lane," 1 for "stay in middle lane," 2 for "go to right lane," 3 for "speed = 5," 4 for "speed 10," 5 for "speed 20," and 6 for "speed 30." The vehicle took a mix of all these actions initially, but in the final 1000 episodes, it mostly adjusted its speed and seldom changed lanes. This shows that the model has room for improvement in this aspect.

Conclusion

In conclusion, our project focused on utilizing deep reinforcement learning, specifically Deep Q-Networks (DQN), to train an autonomous vehicle in the task of obstacle avoidance. Through extensive training and analysis, we gained valuable insights into the model's learning progress and its behavior in the simulated environment.
During the initial episodes, the model faced challenges and incurred penalties due to crashes, resulting in low rewards. However, as the model continued to learn and adapt, it gradually improved its performance and received positive rewards for maintaining its course without collisions. Despite this progress, the vehicle encountered difficulties in navigating without crashing after a certain point, indicating the need for further refinement of the model's hyperparameters and techniques.

Moving forward, there are several avenues for future work and enhancement. Refining the model's hyperparameters, such as the learning rate and exploration rate, could potentially improve its learning efficiency and performance.. Further investigation into fine-tuning the balance between speed and lane changes could enhance the vehicle's navigation and overall efficiency.

In conclusion, our project has laid the foundation for utilizing deep reinforcement learning and DQN in the context of autonomous vehicle obstacle avoidance. While challenges and areas for improvement were identified, the project has opened up possibilities for future advancements in training models that can effectively navigate complex environments and overcome obstacles. By addressing the limitations and incorporating future research directions, we can pave the way for safer and more efficient autonomous driving systems.

References

1. Stens, A., Szewczyk, I., Lindseth, F., & Kiss, G. (2022). AI-agents Trained Using Deep Reinforcement Learning in the CARLA Simulator.
2. Xie, M., Trassoudaine, L., Alizon, J., Thonnat, M., & Gallice, J. (1993). Active and intelligent sensing of road obstacles: Application to the european eureka-prometheus project. In 1993 (4th) International Conference on Computer Vision (pp. 1-6). DOI: 10.1109/ICCV.1993.378154.
3. Sharma, A., & Sharma, S. (2021). WAD: A Deep Reinforcement Learning Agent for Urban Autonomous Driving. [Online]. Available: https://arxiv.org/abs/2108.12134.
4. Sutton, R. S., & Barto, A. G. (1999). Reinforcement learning. Journal of Cognitive Neuroscience, 12(2), 209-239.
5. Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y., & Zhokhov, P. (2017). OpenAI Baselines. Retrieved from https://github.com/openai/baselines.
6. Dworak, D., Ciepiela, F., Derbisz, J., Izzat, I., Komorkiewicz, M., & Wójcik, M. (2019). Performance of LiDAR Object Detection Deep Learning Architectures Based on Artificially Generated Point Cloud Data from CARLA Simulator. In Proceedings of the 2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR) (pp. 600-605). Miedzyzdroje, Poland.
7. Marti, E., de Miguel, M. A., Garcia, F., & Perez, J. (2019). A Review of Sensor Technologies for Perception in Automated Driving. IEEE Intelligent Transportation Systems Magazine, 11(2), 94-108.
8. Sutton, R., & Barto, A. (2018). Reinforcement Learning: An Introduction (2nd ed.). The MIT Press.
9. Terapaptommakol, Wasinee, Phaoharuhansa, Danai, Koowattanasuchat, Pramote, & Rajruangrabin, Jartuwat. (2022). Design of Obstacle Avoidance for Autonomous Vehicle Using Deep Q-Network and CARLA Simulator. World Electric Vehicle Journal, 13, 239. https://doi.org/10.3390/wevj13120239.