Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

SAFE AND EFFICIENT LANE CHANGING IN AUTONOMOUS VEHICLES USING ARTIFICIAL INTELLIGENCE

SAKSHAM AGARWAL

April 19th, 2022

Supervisor: Dr. Mélanie Bouroche

A dissertation submitted in partial fulfilment
of the requirements for the degree of
MAI (Computer Engineering)

Declaration

I hereby declare that this dissertation is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at `http://www.tcd.ie/calendar`.

I have also completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at `http://tcd-ie.libguides.com/plagiarism/ready-steady-write`.

Signed: Saksham Agarwal Date: 19th April, 2022

Abstract

Autonomous vehicles are now a part of daily life. While considering hard constraints such as traffic rules and collision avoidance, an autonomous driving system should be able to maximize comfort, safety, and efficiency. The primary factor in 94 percent of all fatal crashes is human error. So, reassuringly, greater use of autonomous vehicles could limit humans' mistakes and eliminate millions of otherwise avoidable deaths. With an increase in traffic congestion, air pollution also increases. Object detection and object classification algorithms allow self-driving cars to recognize things, comprehend circumstances, and make decisions. Better ML models are needed to improve their safety and motion planning. The primary goals and objective of this project titled' Safe and efficient lane changing in Autonomous vehicles using Artificial Intelligence' is to implement Artificial Intelligence techniques to improve the safety in which Autonomous vehicles change lanes, thus reducing accidents and improving the motion planning, taking into account the different surrounding factors that affect the trajectory of Autonomous vehicles. Through the use of SUMO urban mobility software, this dissertation will implement Deep Reinforcement learning methods in a simulation environment using Q-learning methods like Deep Q Networks and Policy optimization Reinforcement Learning methods like Advantage Actor-Critic (A2C) and Proximal Policy Optimization(PPO), and the results of these methods are evaluated using different parameters. Reward parameters are defined, including comfort, safety, and efficiency. These three metrics allow us to implement a solution where an ego vehicle can make a lane-changing decision from an action space in a reinforcement learning environment in a mandatory lane-changing scenario. After evaluating the learning performance of different RL methods mentioned above, it was deduced that Policy Optimization-based Reinforcement Learning models tend to perform better than Q learning methods. Policy-based RL methods tend to learn the value of the policy by steps of exploration in the Reinforcement Learning environment. In contrast, Q learning methods learn independent of the agent's action in the environment. New Q learning methods like Double DQN and DQN can improve their performance compared to DQN being used without any add-ons. The results and findings and the implementation of the approach are further discussed in the report.

Acknowledgements

I would firstly like to express my heartfelt gratitude to Dr. Mélanie Bouroche (Assistant Professor, School of Computer Science and Statistics) for her constant support and providing guidance throughout the duration of this project. Her willingness to lend her time and expertise so generously has been very greatly appreciated, and without her feedback and assistance, the completion of this final year dissertation would not have been possible.
I would also like to thank my parents for supporting me and motivating me throughout this project, without which I would not have been able to complete it.

Contents

List of Figures

List of Tables

1 Introduction

Autonomous vehicles are now a part of daily life. Different assistance technologies, such as adaptive cruise control (ACC) or traffic-jam help, are already available from car manufacturers. On public highways, test vehicles with various levels of automation are already being tested. While taking into consideration hard constraints such as traffic rules and collision avoidance, an autonomous driving system should be able to maximize comfort and safety. Highways are highly dynamic environments, making it impossible to predict all state evolutions involving the activity of other agents and limiting predictions to the short term. The vehicle, on the other hand, can collect data from its previous experiences and should use this data to better its decision-making strategy.(1)

1.1 Context and Research Motivation

For autonomous vehicle decision-making, machine learning techniques have outperformed several rule-based systems. Machine learning is difficult to implement due to the risk of performing harmful actions and slow learning rates. Many of these concerns should be addressed by offering a reinforcement learning-based technique paired with formal safety verification to verify that only safe actions are selected at all times. Some of these techniques are studied in this research project to propose an effective solution to make autonomous vehicles safe.

Due to the influence of surrounding traffic participants, traffic rules, and inconsistent optimization requirements, motion planning for autonomous vehicles is difficult. Separating the planning task into high-level decision-making and maneuver implementation is one way to deal with this complexity. High-level decision making involves building a system of rules and deducing the optimal maneuver online.(2)

According to current trends, object recognition and detection are considered one of the most significant tasks of Autonomous Vehicles. One such use is traffic sign recognition, commonly employed in automotive systems such as Advanced Driver Assistance Systems (ADAS) and,

more recently, in autonomous vehicles. However, because of road situation's complex and dynamic nature, it faces numerous problems in its detecting process. This is because of limitations due to camera Angle of View(AOV). There might be instances where road signs are not present in different conditions resulting in camera impairment, thereby affecting the ability of the AV to detect road signs timely and accurately.(3) (4)

The advancement of computing power and the increased availability of data have prepared the way for the application of machine learning approaches to decision-making in recent years. Machine learning allows vehicles to learn from data and refine their learning strategy, and one such technique is Reinforcement Learning. Creating a simple yet adequate depiction of the environment can greatly speed up the learning process. Furthermore, ensuring that the vehicle only executes safe maneuvers, i.e., does not cause a collision, adds to the design's complexity. When performing learning in real traffic with other traffic participants, safety is incredibly important.

1.2 Problem Statement

The prospect of a future where we don't have to drive is very appealing to many.Approximately 1.3 million people die each year due to road traffic crashes and the primary factor in 94 percent of all fatal crashes is human error.(5) So, reassuringly, greater use of autonomous vehicles could limit humans' mistakes and eliminate millions of otherwise avoidable deaths. With an increase in traffic congestion, air pollution also increases.Object detection and object classification algorithms allow self-driving cars to recognize things, comprehend circumstances, and make decisions. Better ML models are needed to improve their safety and motion planning.

1.3 Project Goals

The primary goals and objective of this project titled 'Safe and efficient lane changing in Autonomous vehicles using Artificial Intelligence' is to implement Artificial Intelligence techniques to improve the safety in which Autonomous vehicles change lanes, thus reducing accidents and improving the motion planning taking in account the different surrounding factors that affect the trajectory of Autonomous vehicles.

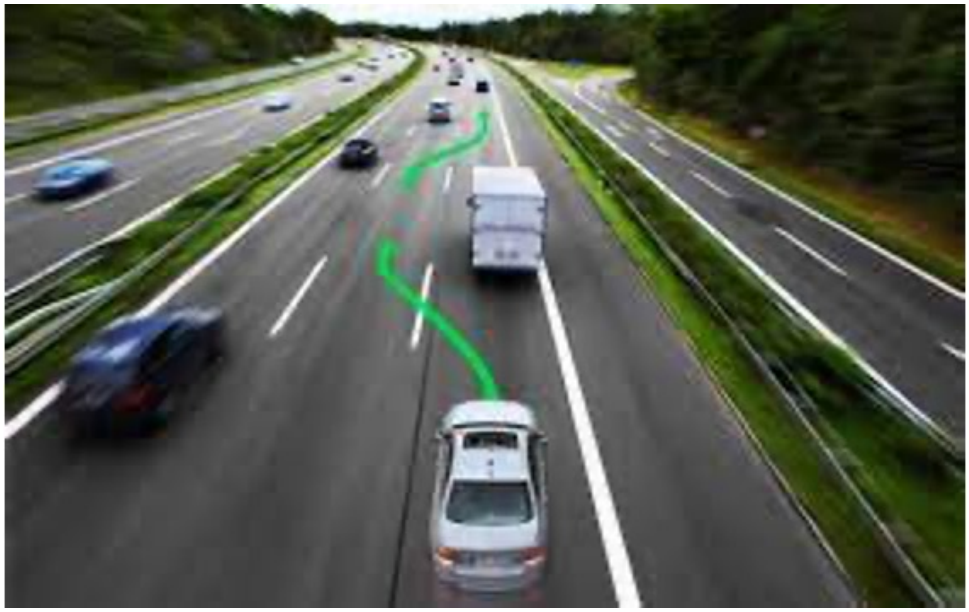

Figure 1.1: Lane Changing in Autonomous Vehicles

1.4 Structure of Report

Here is an overview of how this thesis is structured into the subsequent chapters:-

- Chapter 2 will provide information on related work done in Autonomous lane changing and the background of knowledge required to build the project.
- Chapter 3 will cover the design choices for this project, related technical workings for the proposed design and the experimental methods used for the project
- Chapter 4 will cover the results obtained as part of the ran experiments.
- Chapter 5 will conclude the thesis and talk about any future scope of this project.

2 Literature Review

This chapter gives us context on work done in Safe and Efficient Lane changing using Artificial Intelligence Techniques and helps us understand some concepts related to this project. Different Artificial Intelligence techniques have been used in the past to improve the safety and efficiency of autonomous vehicles, and the subsequent sections in this chapter review the work done on this topic in recent years.

2.1 Lane Changing Behaviour on Freeways

Freeways form an essential aspect of the transportation infrastructure. When traffic exceeds capacity, however, it becomes crowded. As a result, capacity is a critical traffic variable. To address these issues, it would be beneficial if the driver behavior could be anticipated and traffic dynamics could be forecasted. This could be valuable for road design or driver behavior improvement. (3) (6)

2.1.1 Lane Changing Behavior on Multilane Freeways

Most microscopic simulation models start with the desired speed and then adjust the lane if necessary. The most common tiny lane-change algorithms are rule-based and discrete-choice-based models. Different gap acceptability requirements have been considered in rule-based models, taking into account the variability across drivers. Many lane-changing models use multiple parameter sets to account for driver variability. It has been discovered that different methods are used by different type of drivers when it comes to car-following behavior. It has been found that people drive on the freeway using the four tactics listed below. (6)

- Strategy 1: On a freeway stretch, drivers set a preferred speed and maintain it as much as feasible. They shift lanes and overtake if necessary to maintain their momentum. They keep their desired speed during the overtaking attempt as well. This technique is also known as "speed leading".

- Strategy 2: Drivers set the desired speed and maintain it as much as possible, much like in the previous approach. They might switch lanes to pass a slower car. In contrast to the speed leading technique, the drivers increase their speed while overtaking rather than maintaining the original desired speed. This tactic is referred to as "speed leading with overtaking".
- Strategy 3: Drivers select a chosen lane and adjust their speed to match that lane's speed, although within certain limits (the standard, acceptable range is around 40 km/h). This is known as "lane leading".
- Strategy 4: Drivers "go with the flow" since they don't have a designated lane or desired speed in mind. This is also known as "traffic leading".

.(6)

2.1.2 Lane Changing Decision Algorithm

One of the autonomous vehicle's most important driving behaviors is lane-changing in a highway scenario. Vehicle lane-changing is primarily used to increase driving performance and safety. The self-driving car uses a multi-sensor fusion algorithm to detect its surroundings, chooses the best route after deciding to change lanes, and then does a risk assessment on the selected course, determining whether the current lane change action meets the safety requirements. The lane shifting action will be performed if the initial conditions are met else, the car's following state will be maintained. (3) (7)

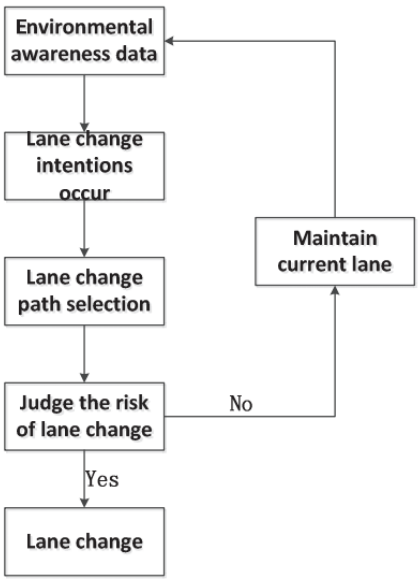

Figure 2.1: Lane changing decision flow chart
(7)

2.1.3 Different types of Lane change

One of the most critical aspects of motorway driving is a lane change. Drivers change lanes for various reasons, including increasing speed or switching into the correct lane before the following turning action downstream. A lane change can be classed as mandatory or discretionary based on the driver's motivation. When a driver wants to move their respective vehicle from its current lane to a target lane for a right or left turn or lane closure, it is known as a Mandatory Lane Change(MLC). A discretionary lane change (DLC) occurs when a driver wants to go faster, have a longer following distance, have a better line of sight, or have a better riding quality in the target lane.(8)

Lane Changing Decision Variables

A lane change can be described as a four-step procedure, whether mandatory or discretionary:-

(1) Motivation;

(2) Selection of target lane;

(3) Checking for the possibility of the movement;

(4) The actual movement;

2.2 Using Reinforcement Learning Techniques

Reinforcement learning is a form of Machine Learning that rewards and punishes desired and undesirable behavior. RL is a machine learning subfield that studies approximate optimum decision-making in natural and artificial systems. A reinforcement learning agent is capable of perceiving and interpreting its surroundings, taking actions, and learning through trial and error.

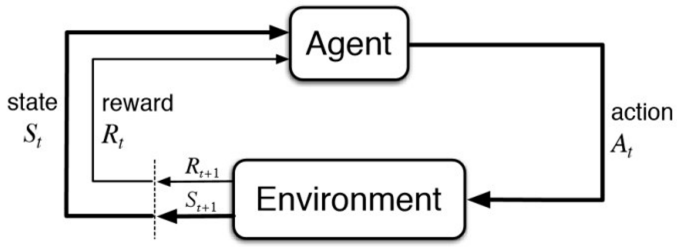

Figure 2.2: RL agent in an Environment Space
(9)

2.2.1 Types of RL algorithms

When talking about Reinforcement learning models, they are mainly categorized into model-based RL and model-free RL algorithms. The primary difference between these categories of models is that the model-free RL algorithms only try to make predictions based on the current state values, whereas the model-based RL algorithms try to make predictions of the future state of the model to try to to generate a best possible action.

These can be seen below in Fig 2.3

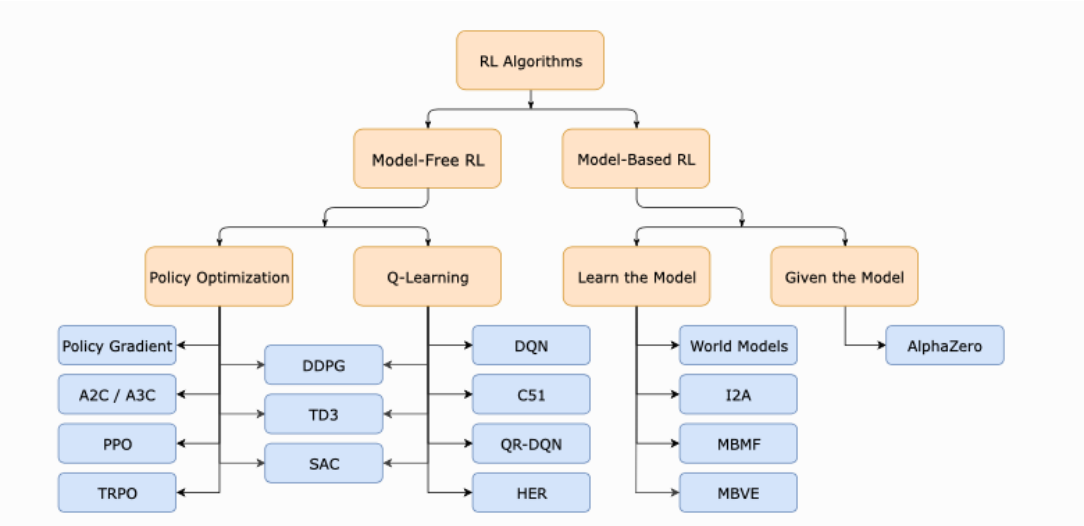

Figure 2.3: Categorization of RL Algorithms
(10)

2.2.2 RL experiments with safety actions

Many machine learning techniques have been developed for Autonomous Vehicles to improve user experiences while maximizing comfort and safety. In 1989, one of the first machine learning algorithms for self-driving cars was proposed. They created an autonomous land vehicle based on neural networks that use camera and laser range inputs to stay on the road. The vehicle Stanley, capable of driving at very high speeds in diverse and unstructured off-road environments using machine learning and probabilistic reasoning for planning, made another substantial contribution to the road following in the DARPA competition. On the other hand, Stanley could only drive in static situations and could not deal with dynamic challenges. (2) (11)

In another experiment,(12) RL was used to operate autonomous vehicles in real-time. For teaching a controller to steer an automobile based on continuous inputs, they used the

Neural Fitted Q Iteration (NFQ) approach(13). However, while the policy can steer the car, it is not ideal.

A case study demonstrated that an agent can learn a low-level control task from scratch using RL. They trained an agent how to maneuver a front wheel to drive as near a predetermined track as feasible. However, they were not able to teach the agent to avoid scene objects in every situation. This research utilized a policy-gradient technique to safely train an agent to follow a front vehicle. The agent learned steering and braking control based on three continuous input aspectsl. Even though it has been proven that learning low-level control works, complex scenarios with several other dynamic constraints remain difficult. Furthermore, there are established control approaches for maneuver execution that can be used instead to concentrate on high-level decision-making. (2)

Branka Mirchevska, Christian Pek, Moritz Werling, Matthias Althoff, and Joschka Boedecker proposed a DQN-based RL approach for safe high-level decision-making for autonomous driving on highways in their paper High-level Decision Making for Safe and Reasonable Autonomous Lane Changing using Reinforcement Learning. (2)

- According to their proposed approach, it teaches how to make high-level judgments while a low-level system executes lane change or lane-keeping on highways with an arbitrary number of lanes.
- use a minimal representation of the environment to decrease the dimensionality of the state space for quick learning, and
- Include safety checks in the system to guarantee that the agent only performs safe operations.

.

Safe actions are defined as a distinct collection of Actions(A) that include the following:

- A1: Perform the lane change to the left
- A2: Stay in the same lane
- A3: Perform the lane change to the right

.

The RL agent aims to get as close as possible to the specified velocity. On the other hand, different driving objectives can be included. This system provides collision-free learning even in real traffic by adding a computationally efficient safety verification tool. Researchers were

able to train a small neural network and demonstrate the effects of their learning strategy by comparing the RL agent against an existing rule-based technique. They chose A(Actions) because they wanted to train the RL agent to make high-level decisions in highway settings, and these actions are the bare minimum for that. After selecting one of these actions from the RL agent at a particular time step t, the safety system evaluates the action's safety. If the action is safe, the underlying system handles the low-level execution. (2)

2.2.3 Using Deep Q Network Technique

To approximate the Q state-action value function, they adopt the Deep Q Network (DQN) technique (13), which is based on Q-learning but uses the power of neural networks. Batch mode reinforcement learning is a sub-field of dynamic programming-based reinforcement learning. It was initially used to describe a version of RL in which the agent is provided a batch of transition samples from which to build the best policy.

To approximate the Q state-action value function, they adopt the Deep Q Network (DQN) technique (13), which is based on Q-learning but uses the power of neural networks. A sub-field of dynamic programming-based reinforcement learning was used in this experiment to describe a version of RL known as batch mode reinforcement learning. In this the agent is provided a batch of transition samples from which to build the best policy. These steps can be depicted in the following Figure 2.4:-

Figure 2.4: Steps used in DQN approach
(2)

- Collection of data samples
- In order to learn the best feasible policy from the source data, the batch mode RL Algorithm is used. and
- Using the learned policy to solve a specific problem.

(2)

The neural network layer can be seen in the Figure 2.5 below

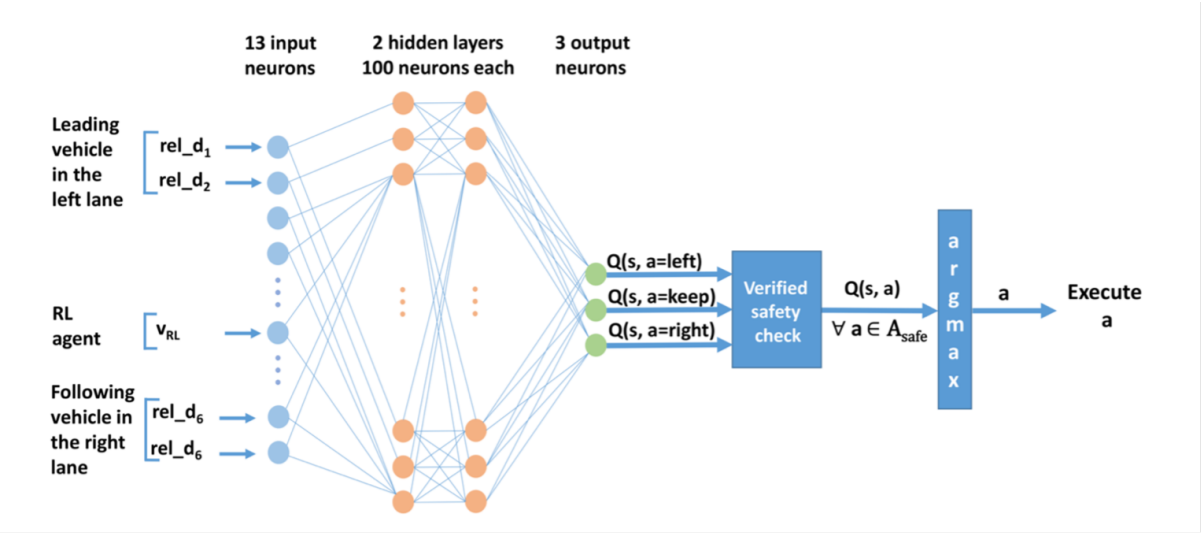

Figure 2.5: Neural network layer
(2)

This RL-based technique for autonomous vehicles in safe and efficient lane changing shows that it can achieve high velocities closer to the intended speed limit while never causing an accident. It outperforms the baseline model rule-based agent. Other related works in Intelligent vehicle Automatic Lane Changing include a Lateral Control Method based on Deep Learning and Computer Vision for traffic signal and lane identification in autonomous vehicles to ensure driver safety and reduce road accidents. (2) (14)

2.2.4 Using Proximal Policy Optimization Deep RL method

While various rule-based solutions for solving lane change problems for autonomous driving have been developed, their effectiveness is often limited due to the inconsistency and complexity of the driving environment. Deep reinforcement learning (DRL) has demonstrated potential performance in many application fields. Therefore machine learning-based methods offer an option. The study's researchers present an automated lane change method based on proximal policy optimization and deep reinforcement learning in this paper, which shows significant gains in learning efficiency while maintaining stable performance. In difficult situations, such as dense traffic, the trained agent can develop a smooth, safe, and efficient driving policy to make lane-change judgments. The proposed policy's effectiveness is demonstrated using task success and collision rate indicators.(15)

The simulation findings show that lane change techniques may be learned quickly and done safely, smoothly, and efficiently. Deep RL algorithms' implementations of lane change techniques are frequently hampered by their poor learning rates. This difficulty is solved in (2) and talked more about in detail in 2.2.3 by employing a minimum state representation

with only 13 continuous features, which allows for a faster learning rate when training a DQN. While these methods are focused on modifying the feature space, more effective RL algorithms can also be used to speed up learning rates and reduce policy learning variation.

Although deep reinforcement learning approaches can maximize the intended reward, they do not always ensure safety during learning and execution. To overcome the issues mentioned above, the researchers propose a deep reinforcement learning method based on safe proximal policy optimization (PPO), which incorporates the policy with a safety intervention module in this paper. Using PPO-based deep reinforcement learning, this work aims to build a decision-making technique to enable automatically required lane change maneuvers with the goals of safety, efficiency, and comfort.(15)

The above mentioned goals can be defined below as:

(1) Comfort: Evaluation of Jerk

(2) Efficiency: Evaluation of least time required to travel the maximum distance

(3) Safety: Determining the likelihood of crashes and near-collisions.

Reinforcement learning can train an agent how to interact with its surroundings to maximize the predicted cumulative rewards for a particular task. The value-based and policy-based methods are the two types of RL algorithms. While value-based approaches can use neural networks to approximate the value function in an off-policy way, policy-based methods can directly maximize the quantity of advantage while remaining stable during function approximations. As a result, the focus of this research is on policy-based RL approaches.(15)

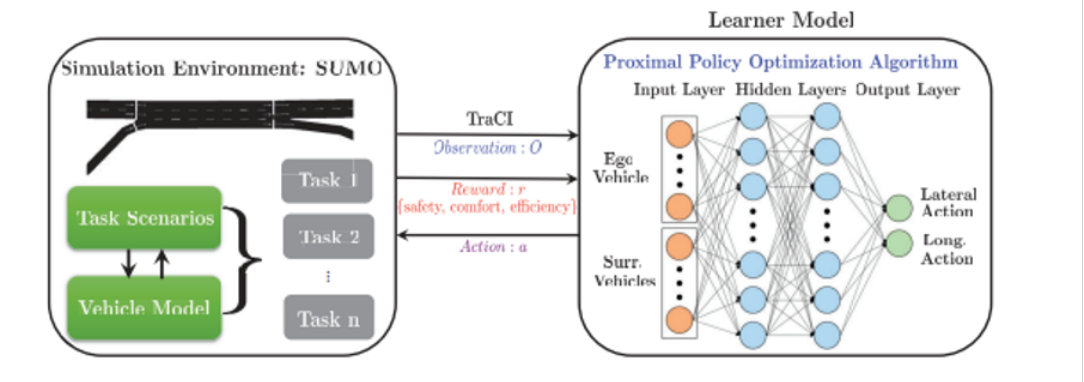

Figure 2.6: Architecture of PPO lane change method (15)

The system has two primary components: a learner model and a simulation. The learner model, in particular, use PPO to teach the ego-vehicle (agent) a high-level policy for decision-making tasks while interacting with the surrounding traffic. The simulation environment, which comprises the road network, traffic, and various task scenarios, is created using the SUMO (Simulation of Urban Mobility) (16) high-fidelity microscopic traffic simulation suite, and it is utilized to interface with the training agent. We can access vehicle information in the road network, perform high-level decisions in the learner model, and take into consideration vehicle dynamics created in the simulation model using SUMO and its accompanying traffic control interface (TraCI).

The ego-vehicle receives its current state and the state of its surrounding vehicles from the SUMO environment through TraCI, and these states are passed through the policy network to enable safe, smooth, and efficient driving behaviors on highways. The ego-vehicle then uses the created policy network to determine high-level lateral and longitudinal actions, which it then transmits back to SUMO to simulate the vehicle's path in the next time step and compute the relevant reward.(15)

2.3 Using Computer Vision Techniques

The researchers of this paper apply computer vision to track road surface markings and provide AVs with an additional layer of data for making decisions. They used YOLOv3 to train their detector to recognize 25 classes of road markings. Over 25,000 images were used in this experiment, which demonstrated a robust performance in terms of accuracy and speed of detection. The results will consolidate the traffic sign recognition system, ensuring better reliability and safety throughout AV operations. A new algorithm using Deep Learning technology in Artificial intelligence (AI) application is implemented and tested successfully.(17)

A computer vision-based road surface marker identification system was used as an additional data source for AVs to make judgments in this approach. The system is a Google Collaboratory cloud service-based custom trained YOLOv3 object detector that employs Darknet detections. Using over 25,000 images, the detector was taught to recognize 25 different road surface markings. Darknet is an open-source neural network framework written in C and CUDA and maintained by Joseph Redmon. It is quick to set up and supports both CPU and GPU computing. YOLO (You Only Look Once) is a bounding box regression heads and classification technique object identification algorithm. It is a real-time object detection model based on convolutional neural networks.(18) Its structure is made up of S$\times$S grid cells, including classifiers and regressors.

2.3.1 Implementation and Experiment

For the detector to be properly taught to detect objects, deep learning methods require a good dataset of images and labels. Gathering and labeling the images is part of the data set preparation procedure. Researchers used Google images to download approximately 25,000 images of various road surface markers for the data collection. To train a roust classifier, they used photos with a variety of backdrops and lighting conditions and random objects and the appropriate road surface markers. The road surface markers are partially obscured in some of the photographs, some road surface marks overlap with others, and some are halfway in the frame.

An automation tool called Labeling was used to label the images. Using the Labeling tool, the researchers annotated all the images with the YOLO data set structure and generated a corresponding XML file for each image. The data set were then divided into two parts: 90 percent of the images are set for training, while the remaining 10 percent are set for testing. In the data set, road surface marks were divided into 25 categories.

They used pre-trained convolutional weights on ImageNet for the training. Using these weights as beginning points allows the network to learn more quickly. Weights from the darknet53 mode were used.

The yolov3.cfg file, which comes with the darknet code, is used to train the YOLO dataset, a demo configuration file. This configuration file contains the training parameters. They set the following variables to match the needs of the detector in order to train the road surface marks dataset efficiently. The neural network weights are iteratively changed during the training phase based on the number of mistakes made while training the dataset. They used a small subset of the images in one iteration because it was impossible to use all of the images in the training set at once while updating the weights. This is referred to as the batch size. The batch parameter specifies the size of the training batches. They chose a batch size of 64, which means that 64 images are used in one iteration to update the neural network's parameters. The researchers also set the subdivisions to 16, which allows for processing the fraction of the batch size at one time in the used GPU. The learning rate parameter controls how aggressively the neural network learns based on the provided batch of data. It is usually a number in the range of 0.01 to 0.000. The lesser the information, the higher the rate, and as the neural networks see more data, the weights need to change less aggressively. It, therefore, needs to decrease over time. This decrease in learning rate is specified in the configuration file as the policy and steps.(17)

2.3.2 Results and Evaluation

The trained data was tested with 10 percent of images,i.e., 2500 images, which resulted in a mean average precision score of 70.07 percent. The network performance was evaluated using seven parameters: Precision, mean Average Precision (mAP), Recall, F1 Score and Quality. Each of the terms are defined below:-

$$\text{Precision} = \frac{\mathit{TruePositive}}{\mathit{TruePositive} + \mathit{FalsePositive}} \tag{1}$$

$$\text{Recall(Sensitivity)} = \frac{\mathit{TruePositive}}{\mathit{TruePositive} + \mathit{FalseNegative}} \tag{2}$$

$$\text{F1 Score } = 2 * (\frac{\mathit{Precision} * \mathit{Recall}}{\mathit{Precision} + \mathit{Recall}}) \tag{3}$$

$$\text{Quality} = \frac{\mathit{TruePositive}}{\mathit{TruePositive} + \mathit{FalsePositive} + \mathit{FalseNegative}} \tag{4}$$

A number of metrics can be deduced from the above equations from which a number of conclusions can be drawn.These are True and False Positive values, Precision and map scores, f1 scores, quality and False Negative scores.The Convolutional Neural Networks Performance results from the above study is shown below in Table 2.1.

Table 2.1: Performance Results

Parameter	**Value**
True Positive(TP)	2408
False Positive(FP)	42
False Negative(FN)	18
Precision	98.28%
MeanAveragePrecision(mAP)	70.07%
Recall(Sensitivity)	99.25%
F1 Score	98.50%
Quality	97.56%

3 Methodology

This chapter gives us context on the design choices made in this project. The implementation's functional approach will be studied, and different types of Reinforcement learning models used in the proposed solution will be discussed. The tools and simulation environments and key terms related to them will also be discussed in detail in this chapter.

3.1 Introduction

Using Deep Reinforcement Learning algorithms, this project tries to teach a car how to self-drive on the road. The simulation will be carried out using the SUMO traffic simulator(16) and TraCi, a Python module. On a route where overtaking is done via lane change, we examine a stretch of highway with a specific number of lanes and a certain number of cars following each other. We will concentrate on one car that we have control over which will be our RL agent. Using the Deep RL, the automobile must decide whether or not to change lanes. It has the option of going left, right, or staying in the same lane. We will create an environment using Open AI gym (19) which is a toolkit , used primarily for Reinforcement Learning research.

3.2 Design

In the domain of Autonomous Driving using Reinforcement Learning techniques, the car, which acts as the primary agent, needs to have an environment and learn from its surroundings factors, including other cars and different lanes, to perform better actions depending on the problem it is trying to solve. In the case of Lane changing, the car needs to learn from its previous actions and choose whether to stay on the current lane, make a lane change toward the right or make a lane change toward the left while also maintaining its optimal speed. For this purpose, a three-lane track was created using Sumo and Netedit. Using the TraCi python module, the parameters for the simulation can be set and modified.

The simulation track can be seen below in Fig 3.1.

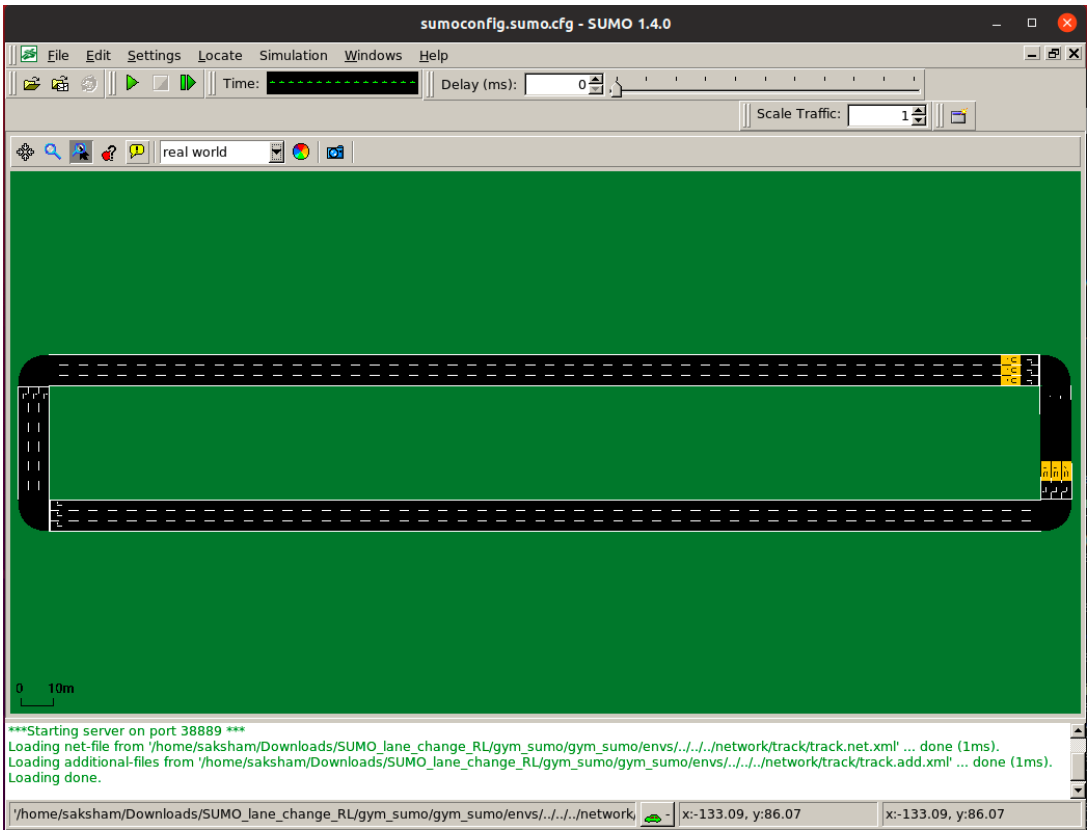

Figure 3.1: A three Lane Design created with NetEdit

3.3 OpenAI Gym Environment

Reinforcement learning (RL) deals with making decision sequences. Reinforcement learning implies the presence of an agent in a given environment. The agent performs an action at each phase, and the environment observes and rewards it. As the agent interacts with the environment, an RL algorithm aims to maximize some measure of the agent's total reward. The environment is characterized as a stochastic Markov decision process in the RL field. The episodic context of reinforcement learning, in which the agent's experience is split into a sequence of episodes, is the emphasis of OpenAI Gym.The agent's initial state is sampled at random from distribution in each episode, and the interaction continues until the environment achieves a terminal state. In episodic reinforcement learning, the goal is to maximize the total reward expectation per episode while achieving a good performance in as few episodes as possible. (19)

It can be seen from Fig 2.2 that we have a learning agent that reacts to its surroundings. A numerical reward from the environment accompanies the new state. As a result, the agent

observes the new condition and acts once more. It is rewarded once more and advanced to the next state. This reward informs the agent if the action was positive or negative. For example, if the agent is playing a game and their scores improve after acting, the reward favors the agent and motivates it to repeat the action in the same situation. If the agent's score drops, it is penalized for not repeating the activity. The cycle continues, and the agent finally learns the dynamics of the environment after many trials and errors. The pattern repeats, and after many trials and errors, the agent finally learns the characteristics of the environment and aims for ever-higher scores, resulting in a practically optimal policy for that environment.

The end performance of an RL algorithm in a given environment can be measured along two dimensions: first, the final performance, which can also be referred to as the average reward for each episode after the training of the model is complete; second, the time required to learn which can also be referred as sample complexity.(19)

Taking reference from the above section 2.2.4, a learner model and a simulation environment are the system's two main components. The learner model, in particular, teaches the target vehicle (agent) to learn while interacting with the traffic around it. A high-fidelity microscopic traffic simulation suite called SUMO (16)is used to create the simulation environment, which comprises the road network, traffic, and various task scenarios. The target vehicle obtains its current state and the state of its surrounding vehicles from the SUMO environment through TraCi. These states are transferred through the policy to enable safe, smooth, and efficient driving behaviors on highways. The target vehicle determines the action based on the developed policy, which is then sent back to SUMO to model the vehicle's movement in the following time step and compute the relevant reward.

The environment was created with a discrete set of Action space which will be talked about in detail in further subsequent sections. There are 4 primary functions in relation to an environment which are called whenever a timestep action is completed:-

(1) env.reset() :- This function resets the state of the environment and obtains initial observations set.

(2) env.render() :- This function helps to visualise the environment

(3) env.step() :- This function applies an action chosen from the action space to the environment. This function will send the action state, whether to stay on the current lane or change the lane to the environment, compute the simulation steps, update the environment parameters, compute the rewards, and return the environment's state, reward, and other information. This is one of the most critical functions in the

implementation.

(4) env.close() :- This function closes down the render frame.

3.3.1 State Space

As seen from Fig 2.2, for an agent to act. It needs to explore the state and then choose from a list of corresponding action spaces. In this environment, the state space comprises a total of 37 features. Out of this vector space, five comprise the RL agent, which includes parameters like distance, speed, acceleration, and position. Both the lateral and longitudinal positions are taken into account for the RL agent. The rest of the feature space is comprised of 8 surrounding vehicles around the ego vehicle. Each surrounding vehicle has four features: speed, distance, acceleration, and lateral position.

3.3.2 Action Space

For the environment, the action space is made up of 3 decisions which can be performed by the RL agent:-

(1) Maintain the current lane

(2) change lane to the right

(3) change lane to the left

3.3.3 Reward function

The computative reward is calculated on the basis of 3 objectives defined in section 2.2.4 which are safety, efficiency and comfort(15). The reward functions for the three objectives are defined below as:-

(1) Comfort: Evaluation of Jerk

$$R_{comf}(t) = -\alpha \cdot \dot{a}_x(t)^2 - \beta \cdot \dot{a}_y(t)^2$$

(2) Efficiency: Evaluation of least time required to travel the maximum distance

$$\begin{cases} R_{time}(t) = -\delta t \\ R_{lane}(t) = -|P_x - P_x^*| \\ R_{speed}(t) = -|V_y - V_{desired}| \end{cases}$$

$$R_{eff}(t) = w_t \cdot R_{time}(t) + w_l \cdot R_{lane}(t) + w_s \cdot R_{speed}(t)$$

(3) Safety: Determining the likelihood of crashes and near-collisions.

$$R_{collision} = \begin{cases} R_{near\_collision} & \text{if } D < d_s \\ -100 & \text{if collision} \end{cases}$$

(15)

In point 1, a_x and a_y are respectively the lateral and longitudinal jerk for comfort reward and α , β are the corresponding weights for them.This incentive mechanism was added to avoid the vehicle's rapid acceleration or deceleration, which could cause discomfort to the occupants. In point 2, R_{lane} is defined as the ego vehicles lateral position with respect to the desired vehicles position and R_{speed} is defined as ego vehicles speeds with respect to desired vehicles speed.In point 3, when computing reward for safety, if the ego vehicles manages to maintain safe distance with respect to other vehicles, it will reward with a value +1 else would reward a negative value -100.

Compute Reward Function

The reward function considers three parameters for which it computes reward values. They are Comfort, Efficiency, and safety. For Comfort, the function computes jerk, which is done by taking the difference between the current speed and target speed, multiplying it by lane width, and then adding a penalty for changing lanes. The lane width is calculated by another utility function that takes in two parameters: the lane id and vehicle name. It returns an integer value representing how wide the lane is on this particular stretch of road to help calculate reward values. There is a third function that takes weights to be applied to these parameter values to calculate rewards, and it returns a tuple with comfort reward, efficiency reward, and safety reward.

3.3.4 Simulation

The simulation aims to get the agent to run in a congested environment without colliding with other vehicles. The target speed of our RL agent(red card) can be modified at which the agent performs a lane change action. Thirty-five autonomous vehicles other than the target vehicle (red car), which also acts as our RL agent, were introduced to the simulation to create moderate traffic to create a virtual traffic scene.The RL agent would be able to vary the maximum speed at which it can perform a lane change. The ego vehicle, which is the RL agent, observes the given state at a particular time S_t and performs an action from a predefined action space at a specific timestep that corresponds to the lane change action.Once this action has been completed, the agent advances to the next state S_{t+1} and is awarded a reward based on the outcome of the previous step. This can be seen in Fig 3.2.

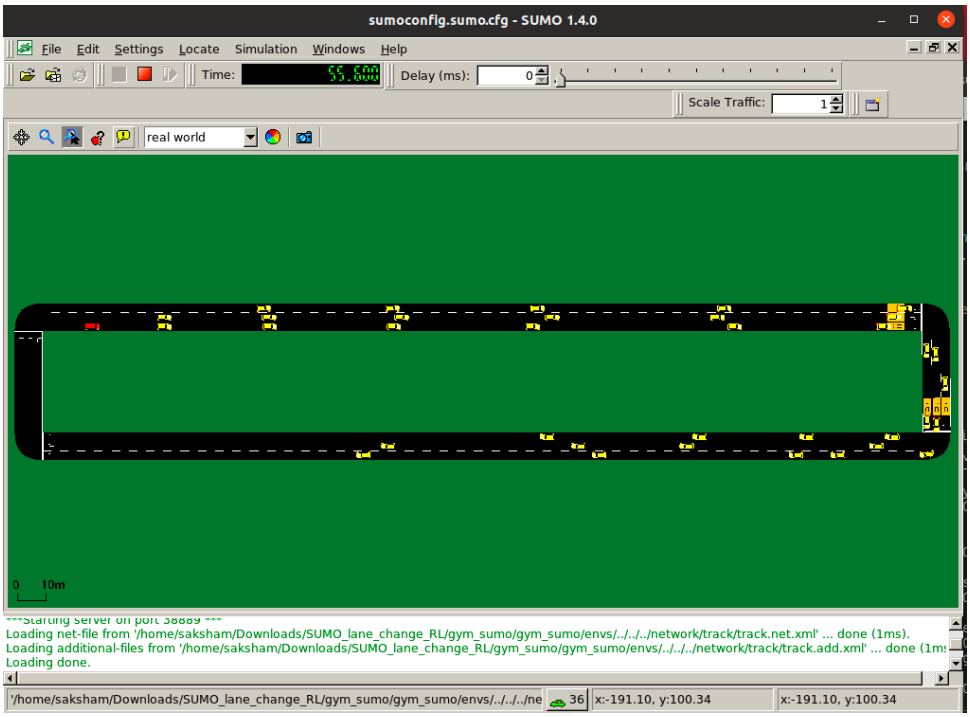

Figure 3.2: RL Agent with surrounding Vehicles

3.4 Implementation

To regulate the lane-changing behavior in Autonomous vehicles in this project, we will employ model-free RL algorithms. Value-based (Q-learning and QL with NNs = DQN) and policy gradient (A2C, PPO) algorithms are two types of model-free algorithms. The goal of value-based RL approaches is to find the best Q-value function. Q-learning is an example of a value-based RL algorithm. In policy gradient, on the other hand, the desired objective

function is directly maximized. Although policy gradient approaches are wasteful in terms of sample size, they are better capable of learning challenging tasks than value-based RL algorithms.Both value-based and policy gradient algorithms are used in this project.

Two other agents are also used in this project to compare the behavior of Reinforcement learning models with constant models. The constant lane change model and a random agent act as a baseline model to compare our results. The constant lane change model will perform a constant lane change action at each timestep. Although this is not a correct measure to check for safety actions when performing a lance change, this is implemented to compare the trained deep RL model performances. A random agent model is also implemented. As the name suggests, it takes a random action from the sample into the environment and acts randomly from the given action space at each timestep. These both are implemented to compare the performances of deep RL models.

3.4.1 DQN

A lot of reference for DQN model has been taken from section 2.2.3 in the implementation.The use of stable-baselines3(20), which is a Reinforcement Learning library for python has been made for the training purposes.

3.4.2 A2C

A2C refers to Advantage Actor Critic method. It is an on policy RL method. To understand about it, we need to take a brief look at A3C method as well.A3C stands for Asynchronous Advantage Actor critic method.A3C uses asynchronous parallel training, in which many workers in parallel settings update a global value function separately. Exploration of the state space is made more effective and efficient using asynchronous actors. A2C, on the contrary, only has one worker and does not support asynchronous functionality. A2C is a policy gradient algorithm that belongs to the on-policy family of algorithms. That is, it is learning the value function for one policy while following it, or, to put it another way, it cannot learn the value function for another policy while following it. Suppose it employs experience replay, for example. In that case, it will apply a different policy as learning from too old data uses knowledge generated by a policy (i.e., the network) that is slightly different from the current state. (20)(21).

3.4.3 PPO

PPO(Proximal Policy Optimization) uses an on-policy approach to train a stochastic policy.It is an on policy algorithm and can be used well with custom environments using discrete action spaces. This means it investigates using the most recent version of its stochastic policy to sample activities. Both the initial conditions and the training technique influence the level of unpredictability in action selection. The policy often grows less random over time as the update rule encourages it to use the benefits it has already discovered. This could lead to the policy becoming stuck in an optimal local state.The Psuedo code for PPO algorithm can be seen below in Fig 3.3.(19)(20)

Algorithm 1 PPO-Clip

Input: initial policy parameters θ_0, initial value function parameters ϕ_0
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
  Compute rewards-to-go $\hat{R}_t$.
  Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
  Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \quad g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

  typically via stochastic gradient ascent with Adam.
  Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

  typically via some gradient descent algorithm.
end for

Figure 3.3: Psuedo code for Proximal Policy Optimization Algorithm (19)

3.5 Difficulties and Challenges faced

Getting the setup of Sumo Environment locally was one of the main issues I faced during this project. SUMO works well on Linux-based platforms, and for a simulation involving multiple surrounding variables, the Reinforcement Learning approaches can take much time. There are also a few limitations and considerations that must be considered when performing RL-based research. Although this project required many complex environment scenarios, Reinforcement learning approaches can be an overkill for more straightforward

problems. Another factor to consider is that RL-based approaches assume that the environment should be Markovian. Finally, Training RL-based models can take much time and computational power and are not always stable. The approximate time to train different RL-based approaches with varying parameters in a vectorized environment took between 8 and 12 hours. Creating a custom Open AI gym environment for the reinforcement Learning-based approach was also tricky. Taking reference from previous work done in this field has helped to implement this project. One more significant issue I faced was the amount of computational power required. The simulations froze in between the experiment randomly, and most of the experiments I conducted using RL techniques were with the Sumo GUI turned off.

4 Results and Evaluation

This chapter will discuss the results obtained from the implementations using the techniques discussed in Chapter 3. Different Plots and graphs for different types of reward functions have been plotted, and learning performance for each RL approach has been identified over a period of time for different metrics like comfort, efficiency, and safety.

4.1 Metrics

The computative reward functions defined above in section 3.3.3 will be evaluated with different RL approaches over a period of time for different metrics like the speed at which the RL agent(the ego vehicle) performs a lane change action. The results will be evaluated keeping the weight parameters for different values as well the network layers constant and only changing the target speed.

4.1.1 Case: Target speed = 100

Since we are taking a highway scenario, keeping the speed 100 seems to be most optimal for testing purposes. Different metrics will be evaluated using different RL approaches. For this case, these are calculated over a period of 20000 timesteps.

Comfort

Figure 4.1: Comfort metric using DQN technique

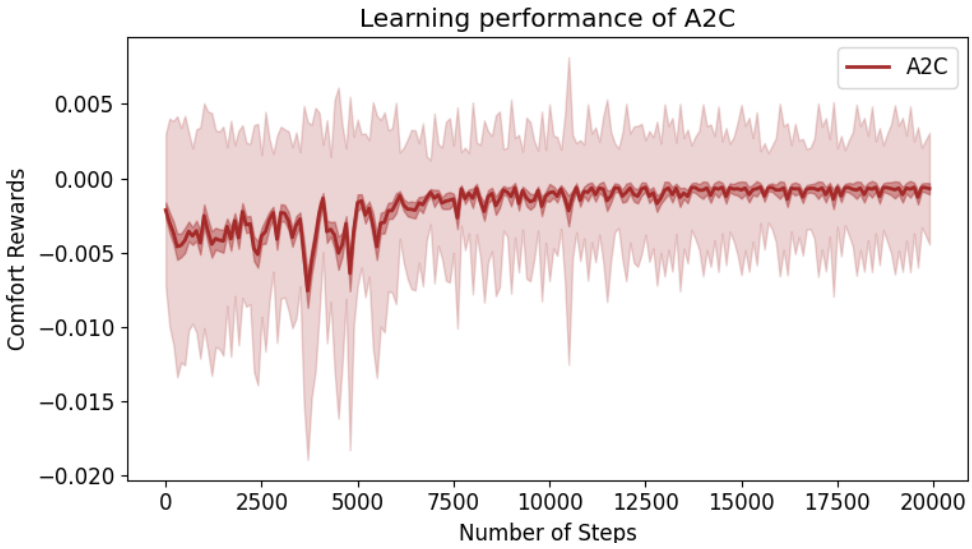

Figure 4.2: Comfort metric using A2C technique

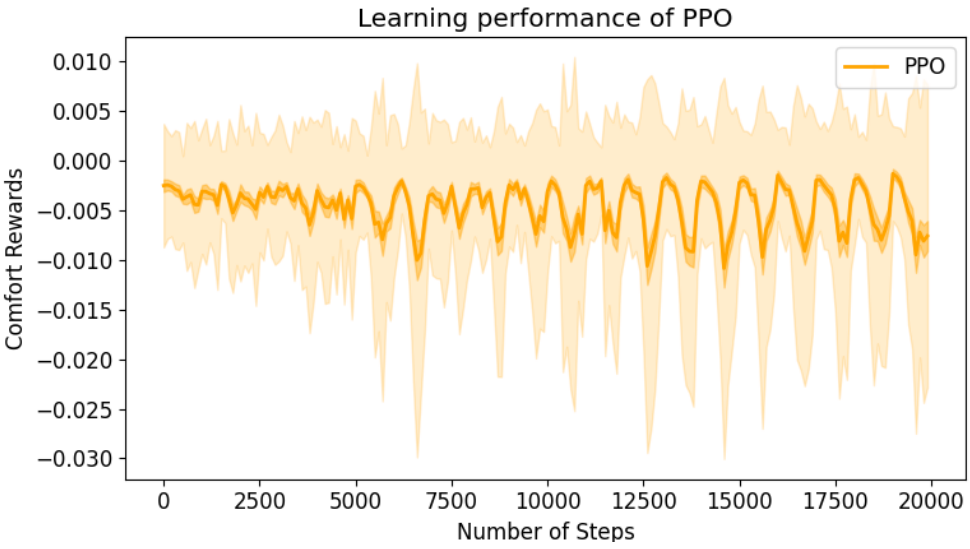

Figure 4.3: Comfort metric using PPO technique

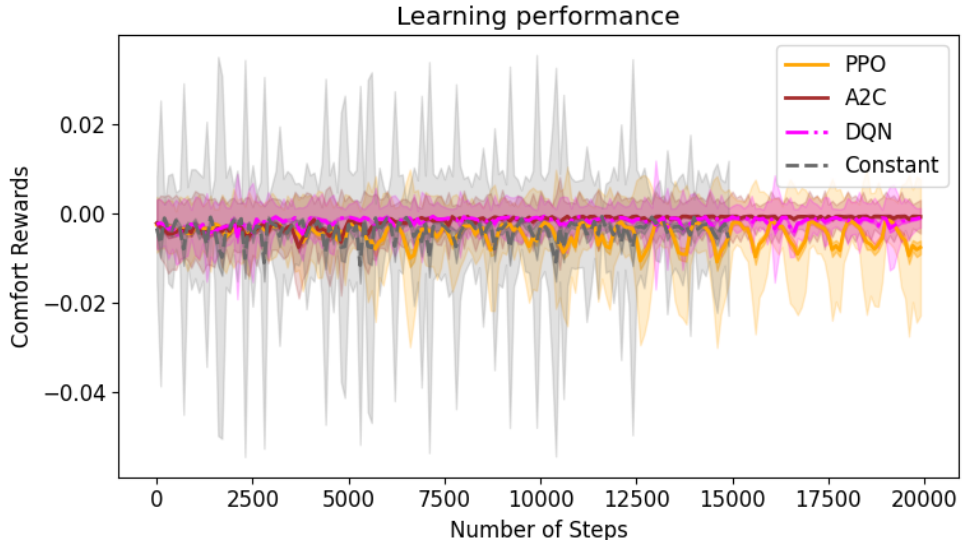

Figure 4.4: Comfort metric performance: RL methods vs Constant Lane Change

It can be seen that for the comfort metric, the learning performance of A2C and DQN tend to perform better than PPO and constant lane change.Comfort metric is used to evaluate jerk, the lower the jerk, the better the performance and comfort will be. As evidently seen from Fig. 4.4. It can also be seen that Proximal Policy Optimization method does not work best for comfort metric and as the timesteps increase, its loss in performance increases. This can be seen in Fig.4.3

Safety

The next metric which would be evaluated is safety. This is one of the most critical evaluation metrics in terms of the research as we are trying to propose a solution where lane changes actions are deemed safe, and the collision is reduced to a minimum.

Figure 4.5: Safety metric using DQN technique

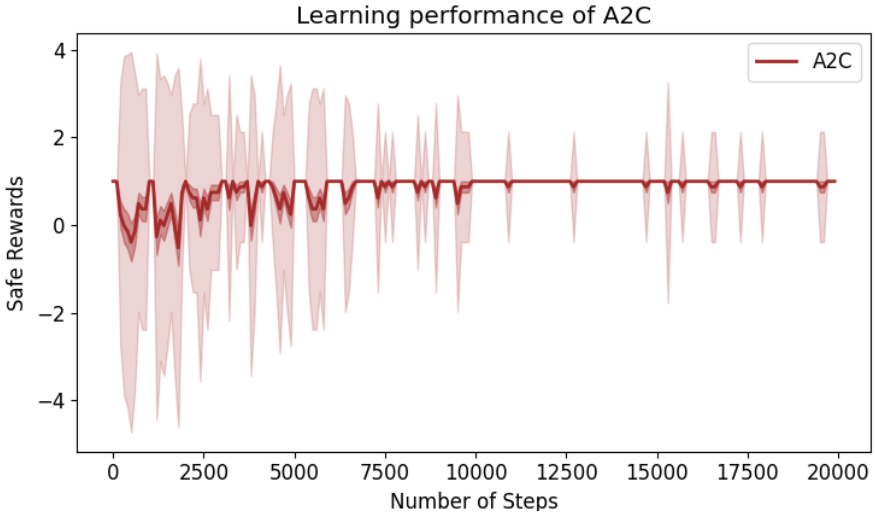

Figure 4.6: Safety metric using A2C technique

Figure 4.7: Safety metric using PPO technique

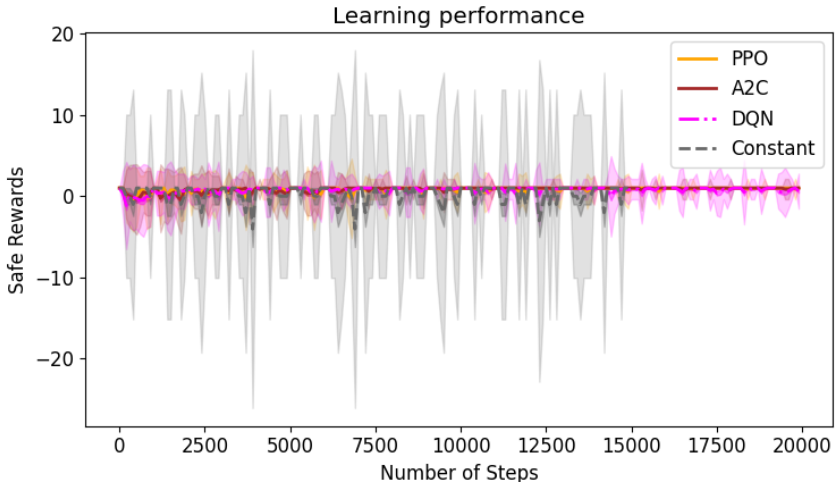

Figure 4.8: Safety metric performance: RL methods vs Constant Lane Change

It can be seen from Fig.4.8 that PPO and A2C models are performing better than DQN. As the model is trained, the loss in DQN performance is increasing, and A2C and PPO are keeping steady. It can also be deduced from Fig. 4.7 that in terms of safety, PPO provides a better convergence rate.But both A2C and PPO models are sensitive to changes.

Efficiency

This metric is used to determine if the RL agent can make safe and efficient lane change actions while also covering the maximum distance in the least amount of time.

Figure 4.9: Efficiency metric using DQN technique

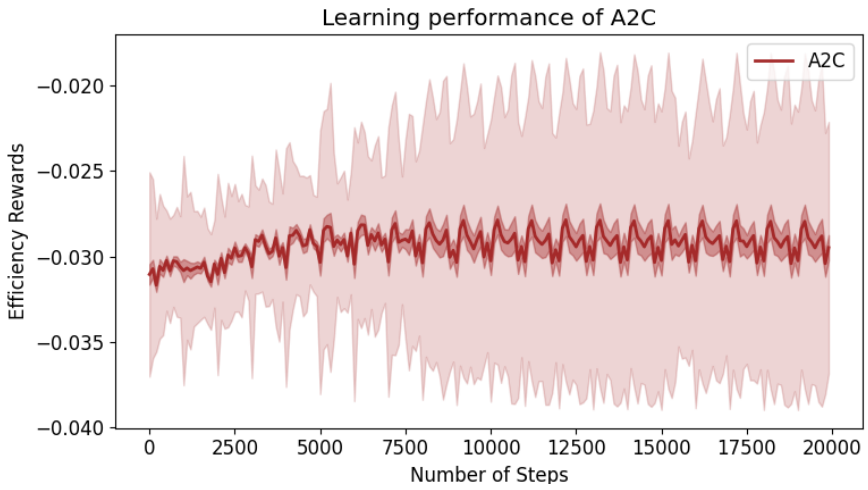

Figure 4.10: Efficiency metric using A2C technique

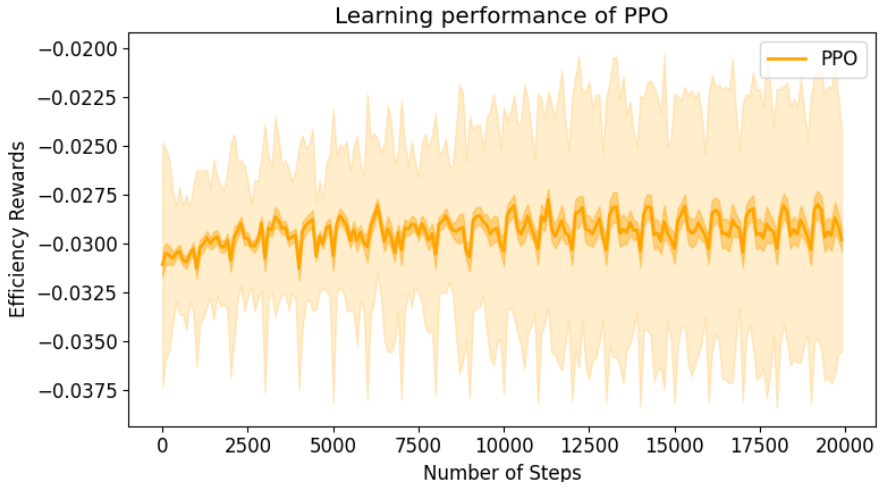

Figure 4.11: Efficiency metric using PPO technique

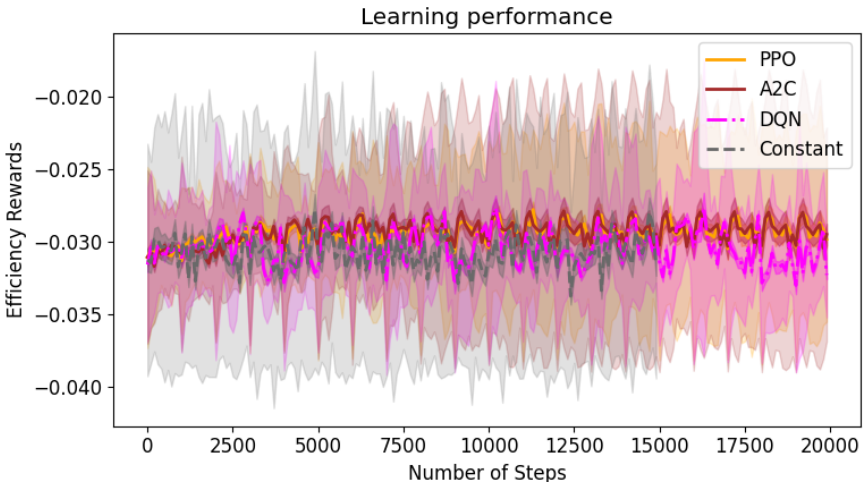

Figure 4.12: Efficiency metric performance: RL methods vs Constant Lane Change

It can be deduced that, when it comes to efficiency, the loss performance of all the models is very high, and nothing particular can be concluded from these graphs in Fig.4.12. Although out of all the models, it can be seen PPO model in Fig. 4.11 seems to be the most steady and stable in terms of loss.Both PPO and A2C seem to be performing than DQN technique.

It can also be seen from the evaluation of all the three reward metrics that in applying in-policy RL methods like A2C and PPO, the learning performance is better than the Q learning-based DQN method. As also learned earlier in the sections of 2.2.3 and 2.2.4, more in-policy-based RL algorithms are being used as they know the value of the policy by steps of exploration in the environment. In contrast, Q learning or off-policy algorithms learn the policy values independent of the agent's actions in the environment. It can be also noted that the poor performance of DQN is because it is not being implemented with any add-ons. DQN alone on Its own is unstable, but new Q learning methods like DDQN and DQN with replay memory can improve its performance.

5 Conclusion and Future Work

This chapter will give context about the work done in this thesis and its key learnings. Essential learnings from this research study will be highlighted, and this chapter will be concluded with a discussion on work to be done on this project in the future.

5.1 Conclusion

This research study provided an opportunity to explore the current evolutions in Autonomous lane changing. Since this is an emerging field and with several innovations happening, it is crucial to propose a solution that can help reduce accidents and collisions in Autonomous vehicles. Efficient Autonomous vehicles can contribute to the environment by reducing air pollution, traffic accidents, and congestion. Many developing countries are now supporting infrastructure for Autonomous vehicles, and these AVs can solve many problems for them.

In this thesis, Chapter 1 gave us an insight into the problem in the current time and introduced the readers to the context of this research topic, and explained to them the need to propose an effective solution in this field.

Chapter 2 talks about the current state of work being done in this field. Exploring what has already been done in this field helped design the solution and present the findings. Exploring the current state of work also made it possible to explore and think about what has not been done and try to implement that.

Chapter 3 introduces the design concepts and choices made in this study. It talks about Action space, State-space, reward functions, and how they are used in the Open AI gym environment. It also talks about the different RL methods used in this project and the choice considerations for using them.

Chapter 4 discusses the results and findings of the project and concludes that policy-based

reinforcement learning methods tend to perform better than Q learning-based methods in the implementation.

Chapter 5 is giving a summary of the work done as part of this research and will talk about future work to be done in this project.

5.2 Future Work

The proposed approach's current implementation only considers a single RL agent that controls the ego vehicle for mandatory lane change scenarios. Future work would include creating multiple RL agents trying to interact with each other, increasing the complexity but improving the decision choices, leading to better lane-changing scenarios.

More emphasis is also needed on the safety check algorithm to ensure the action that the RL agent is taking is safe or not. The current implementation only considers the Euclidean distance and the relative speeds between two vehicles, and the choice is based on it. However, better algorithms need to be formulated to ensure safety checks before making a lane change decision. These are a few things that need to be worked on in the future of this project to make sure the lance changing is perfectly safe and avoid as many collisions as possible, also ensuring that the optimal speed is reached for the vehicle.

Bibliography

[1] Cristina Menéndez-Romero, Franz Winkler, Christian Dornhege, and Wolfram Burgard. Maneuver Planning and Learning: a Lane Selection Approach for Highly Automated Vehicles in Highway Scenarios. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–7, September 2020. doi: 10.1109/ITSC45102.2020.9294190.

[2] Branka Mirchevska, Christian Pek, Moritz Werling, Matthias Althoff, and Joschka Boedecker. High-level Decision Making for Safe and Reasonable Autonomous Lane Changing using Reinforcement Learning. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2156–2162, November 2018. doi: 10.1109/ITSC.2018.8569448. ISSN: 2153-0017.

[3] David Bevly, Xiaolong Cao, Mikhail Gordon, Guchan Ozbilgin, David Kari, Brently Nelson, Jonathan Woodruff, Matthew Barth, Chase Murray, Arda Kurt, Keith Redmill, and Umit Ozguner. Lane Change and Merge Maneuvers for Connected and Automated Vehicles: A Survey. *IEEE Transactions on Intelligent Vehicles*, 1(1):105–120, March 2016. ISSN 2379-8904. doi: 10.1109/TIV.2015.2503342. Conference Name: IEEE Transactions on Intelligent Vehicles.

[4] Suhyeon Gim, Sukhan Lee, and Lounis Adouane. Safe and efficient lane change maneuver for obstacle avoidance inspired from human driving pattern. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–15, 2020. doi: 10.1109/TITS.2020.3034099.

[5] World Health Organization. Road traffic injuries. `www.who.int/news-room/fact-sheets/detail/road-traffic-injuries`, June 2021.

[6] V. L. Knoop, M. Keyvan-Ekbatani, M. de Baat, H. Taale, and S. P. Hoogendoorn. Lane Change Behavior on Freeways: An Online Survey Using Video Clips. *Journal of Advanced Transportation*, 2018:9236028, June 2018. ISSN 0197-6729. doi:

10.1155/2018/9236028. URL https://doi.org/10.1155/2018/9236028. Publisher: Hindawi.

[7] Kangqiang Ouyang, Yong Wang, Yanqiang Li, and Yunhai Zhu. Lane change decision planning for autonomous vehicles. In *2020 Chinese Automation Congress (CAC)*, pages 6277–6281, 2020. doi: 10.1109/CAC51589.2020.9327195.

[8] Matthew Vechione, Esmaeil Balal, and Ruey Cheu. Comparisons of mandatory and discretionary lane changing behavior on freeways. *International Journal of Transportation Science and Technology*, 7, 03 2018. doi: 10.1016/j.ijtst.2018.02.002.

[9] Shweta Bhatt. Reinfocement learning. www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html, June 2018.

[10] Joshua Achiam. Spinning Up in Deep Reinforcement Learning. 2018.

[11] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23 (9):661–692, 2006.

[12] Martin Riedmiller, Mike Montemerlo, and Hendrik Dahlkamp. Learning to drive a real car in 20 minutes. In *2007 Frontiers in the Convergence of Bioscience and Information Technologies*, pages 645–650. IEEE, 2007.

[13] Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European conference on machine learning*, pages 317–328. Springer, 2005.

[14] Li Haixia and Li Xizhou. Flexible lane detection using cnns. In *2021 International Conference on Computer Technology and Media Convergence Design (CTMCD)*, pages 235–238, 2021. doi: 10.1109/CTMCD53128.2021.00057.

[15] Fei Ye, Xuxin Cheng, Pin Wang, Ching-Yao Chan, and Jiucai Zhang. Automated lane change strategy using proximal policy optimization-based deep reinforcement learning. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1746–1752, 2020. doi: 10.1109/IV47402.2020.9304668.

[16] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*, pages 2575–2582. IEEE, November 2018. URL https://elib.dlr.de/127994/.

[17] Irvine Valiant Fanthony, Zaenal Husin, Hera Hikmarika, Suci Dwijayanti, and Bhakti Yudho Suprapto. Yolo algorithm-based surrounding object identification on autonomous electric vehicle. In *2021 8th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, pages 151–156, 2021. doi: 10.23919/EECSI53397.2021.9624275.

[18] Tian-Hao Wu, Tong-Wen Wang, and Ya-Qi Liu. Real-time vehicle and distance detection based on improved yolo v5 network. In *2021 3rd World Symposium on Artificial Intelligence (WSAI)*, pages 24–28, 2021. doi: 10.1109/WSAI51899.2021.9486316.

[19] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[20] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL `http://jmlr.org/papers/v22/20-1364.html`.

[21] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. 2016. doi: 10.48550/ARXIV.1602.01783. URL `https://arxiv.org/abs/1602.01783`.

A1 Appendix

A1.1 Gantt Chart

The Gantt Chart Below summarises the action plan to be followed during this project.

GANTT CHART

PROJECT: AI in Lane Changing
STUDENT NAME. SAKSHAM AGARWAL

Task name	Stard date	End date	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 5	WEEK 6	WEEK 7	WEEK 8	WEEK 9	WEEK 10	WEEK 11	WEEK 12	WEEK 13	WEEK 14	WEEK 15	WEEK 16	WEEK 17	WEEK 18	WEEK 19	WEEK 20	WEEK 21	WEEK 22	WEEK 23	WEEK 24	WEEK 25	WEEK 26	WEEK 27	WEEK 28	WEEK 29	WEEK 30	WEEK 31
PROJECT 1	**13/09/21**	**19/04/22**																														
Exploaration of the topic	13/09/21	27/09/21																														
Research	28/09/21	01/11/21																														
Literature Review	01/11/21	01/02/22																														
Exploring simulation frameworks	15/01/22	15/02/22																														
Model creation	01/02/22	06/03/22																														
Testing and results	06/03/22	01/04/22																														
Writing thesis	15/03/22	19/04/22																														

Figure A1.1: Gantt chart showing the plan of action for MAI project

A1.2 Evaluation of Reward metrics: Target Speed=50

Comfort

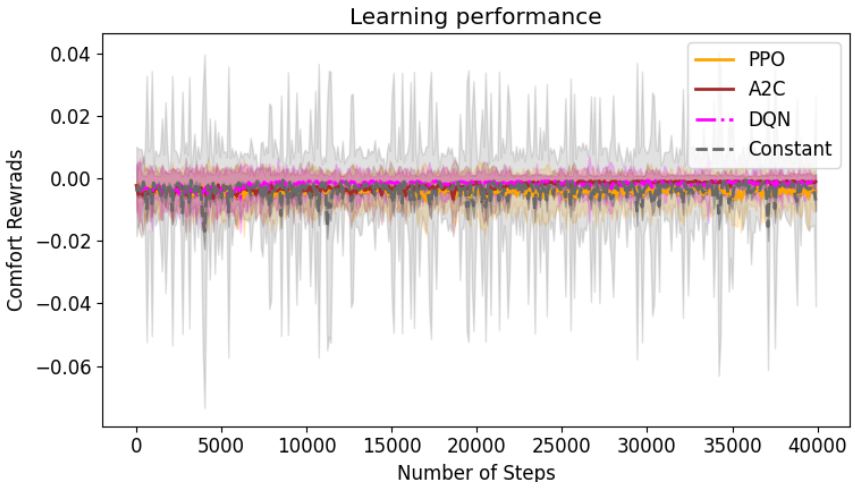

Figure A1.2: Comfort metric performance: RL methods vs Constant Lane Change

Safety

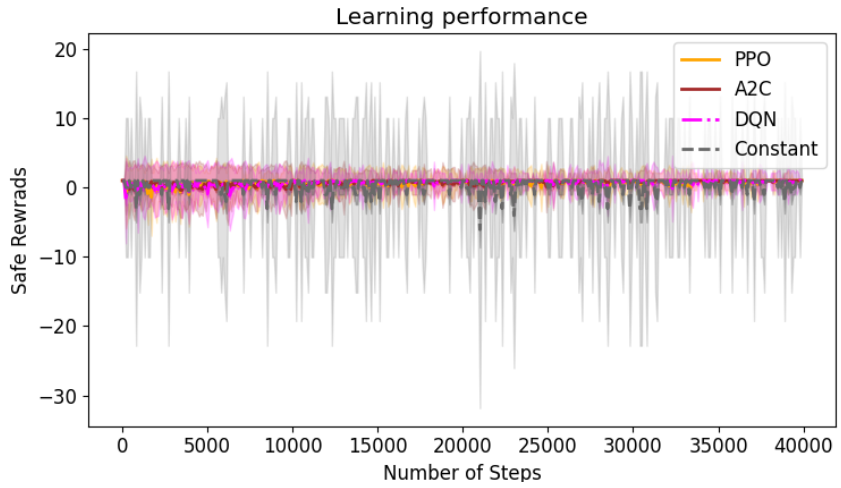

Figure A1.3: Safety metric performance: RL methods vs Constant Lane Change

Efficiency

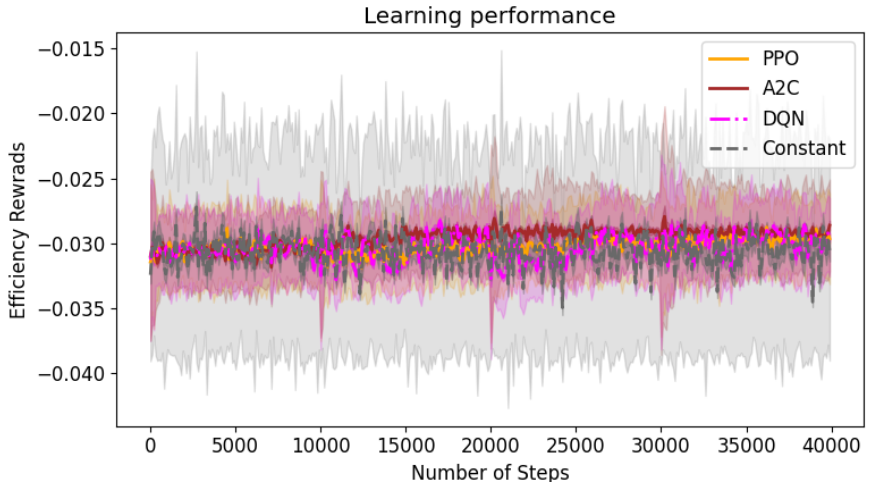

Figure A1.4: Efficiency metric performance: RL methods vs Constant Lane Change

A1.3 Evaluation of Reward metrics: Target Speed=200

Comfort

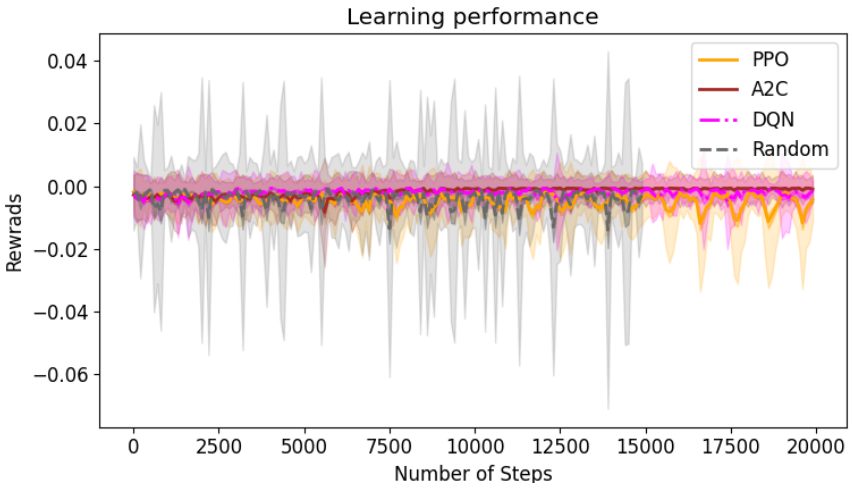

Figure A1.5: Comfort metric performance: RL methods vs Constant Lane Change

Safety

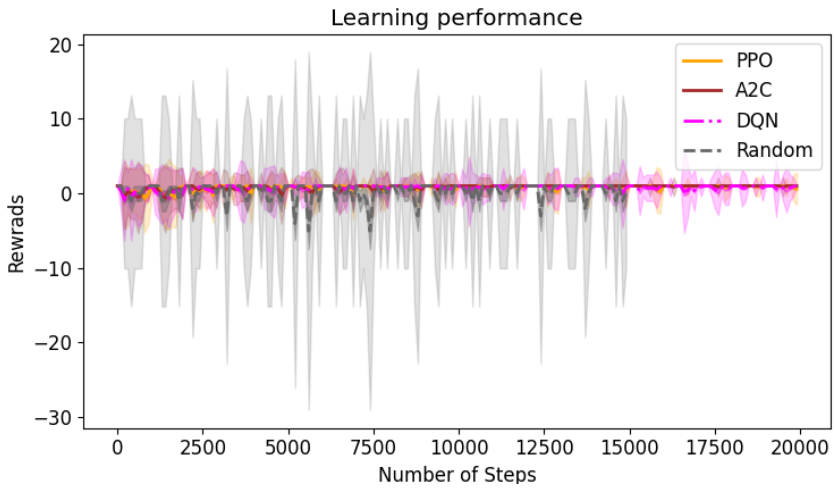

Figure A1.6: Safety metric performance: RL methods vs Constant Lane Change

Efficiency

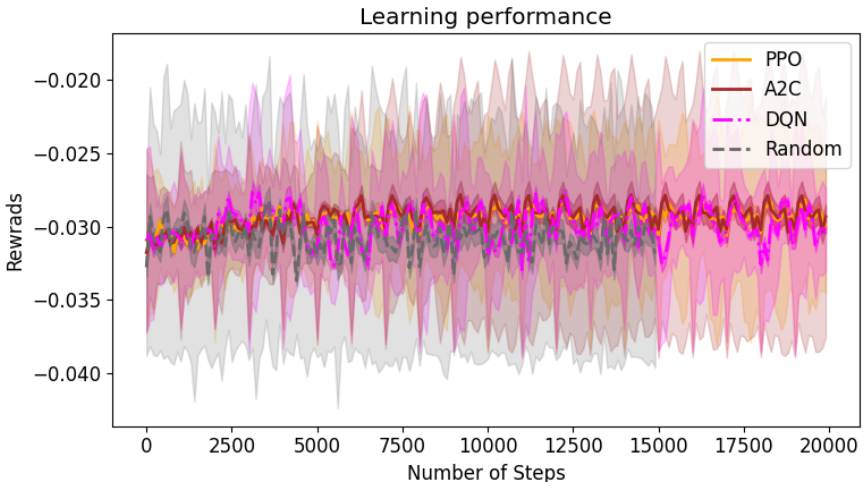

Figure A1.7: Efficiency metric performance: RL methods vs Constant Lane Change

Comfort

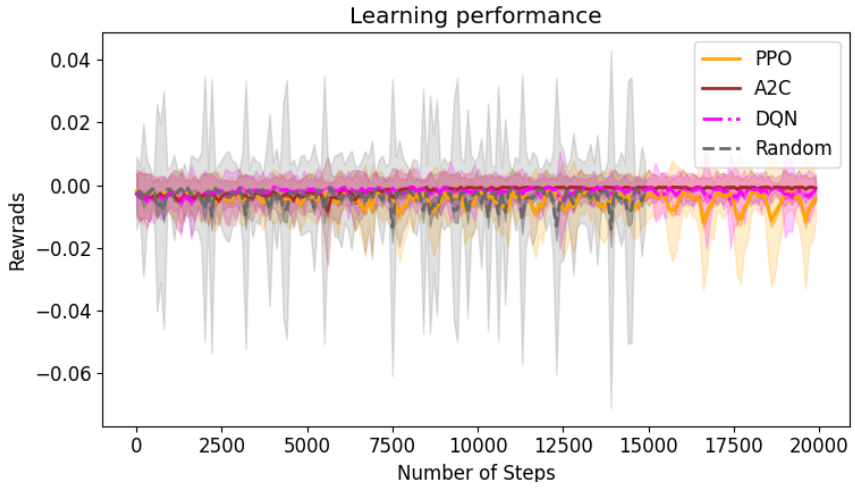

Figure A1.8: Comfort metric performance: RL methods vs Constant Lane Change

A1.4 Some Preliminary experiments done using NEAT

During the initial phases, when I was getting difficulties in setting up the SUMO environment locally and was failing to produce some results, I created two tracks using the python pygame environment. I tried running some simulations for discretionary lane change movements using the NEAT reinforcement learning algorithm in pygame environment.

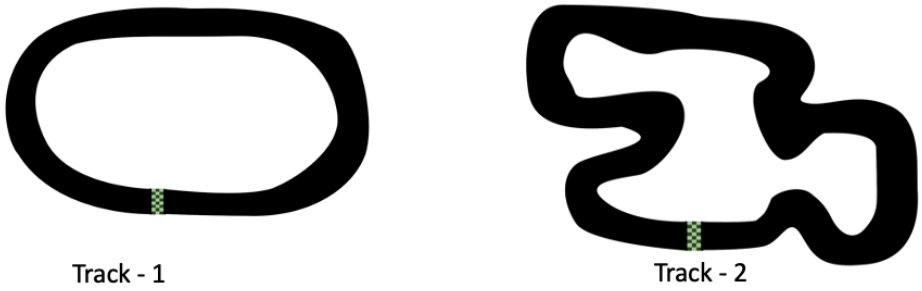

Figure A1.9: Simple tracks created during preliminary experiments

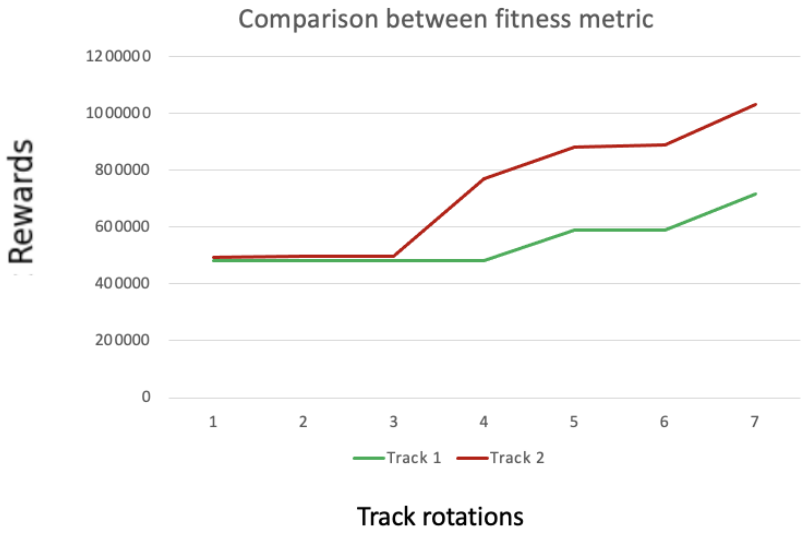

Figure A1.10: Rewards generated using NEAT simulation for both the tracks