UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Autonomous Driving with CARLA

By
Enrique Martínez Martel

Departament de Ciències de la Computació
Universitat Politècnica de Catalunya
Director: Gerard Escudero Bakx

A thesis submitted for the degree of

Informatics Engineering
Major in Computing

Barcelona, June 1st, 2022

This thesis is submitted to the Computer Science Department, Universitat Politècnica de Catalunya in fulfilment of the requirements for the degree in Informatics Engineering
Enrique Martínez Martel, June 2022

Acknowledgements

I would like to thank my parents and brother for the immense support and help during this project. Without their help, completing this project would have been very difficult.

I must also express my gratitude to my teacher, Gerard Escudero Bakx, who gave me the golden opportunity to do this wonderful project on autonomous driving. He guided me throughout all the project and helped me understand many new things. I am thankful to him.

Abstract

The intention of this thesis is to establish a foundation in the world of artificial intelligence, specifically in CARLA, an open-source autonomous driving simulator. The main objective of this work is to enable an autonomous car to drive any given route thanks to the implementation of a Deep Q-Network algorithm.

Throughout the document, it is explained in detail how to create and configure a CARLA environment, including a tutorial on the installation of the software itself with all its dependencies. In addition, an attempt is made to simulate autonomous driving as close to real life as possible using a reward system and an efficient navigation system.

It should be noted that despite having obtained reasonable results and fulfilling the objectives of the thesis, this project is merely introductory, and I hope that it will serve as a basis for future developments.

Resumen

La intención de esta tesis es establecer una base en el mundo de la inteligencia artificial, específicamente en CARLA, un simulador de conducción autónoma de código abierto. El objetivo principal de este trabajo es conseguir que un coche autónomo pueda realizar una ruta cualquiera gracias a la implementación de un algoritmo de Deep Q-Network.

A lo largo del documento se explica detalladamente como crear y configurar un entorno en CARLA, esto incluye un tutorial de la instalación del propio software con todas sus dependencias. Además, se intenta simular una conducción autónoma lo más cercana a la vida real en la que se utiliza un sistema de recompensas y un sistema de navegación eficiente.

Cabe destacar que pese haber obtenido resultados razonables y cumplir los objetivos de la tesis, este proyecto es meramente introductorio y espero que sirva de base para futuros desarrollos.

Resum

La intenció d'aquesta tesi és establir una base al món de la intel·ligència artificial, específicament a CARLA, un simulador de conducció autònoma de codi obert. L'objectiu principal d'aquest treball és aconseguir que un cotxe autònom pugui fer una ruta qualsevol gràcies a la implementació d'un algorisme de Deep Q-Network.

Al llarg del document s'explica detalladament com crear i configurar un entorn a CARLA. Això inclou un tutorial de la instal·lació del propi programari amb totes les seves dependències. A més, s'intenta simular una conducció autònoma el més propera a la vida real on s'utilitza un sistema de recompenses i un sistema de navegació eficient.

Cal destacar que malgrat haver obtingut resultats raonables i complir els objectius de la tesi, aquest projecte és merament introductori i espero que serveixi de base per a futurs desenvolupaments.

Contents

List of Figures

List of Tables

List of Equations

1. Context

This is a Bachelor Thesis of the Computer Engineering Degree, specialisation in Computing, done in the Facultat d'Informàtica de Barcelona of the Universitat Politècnica de Catalunya, directed by Gerard Escudero, doctorate in Artificial Intelligence.

1.1 Introduction

Artificial Intelligence has become extremely popular in today's world. It is the reproduction of regular knowledge in machines that are customised to impersonate the activities of people. These machines can learn and execute human-like tasks. As these technologies continue to grow, they will significantly affect our quality of life.

One of the technologies that have emerged with this growth is autonomous driving. An autonomous car is a vehicle which senses its environment and operates without any human involvement. There are different levels of driving automation [1] as shown on Figure 1, starting from Level 0 (fully manual) up to Level 5 (fully autonomous)

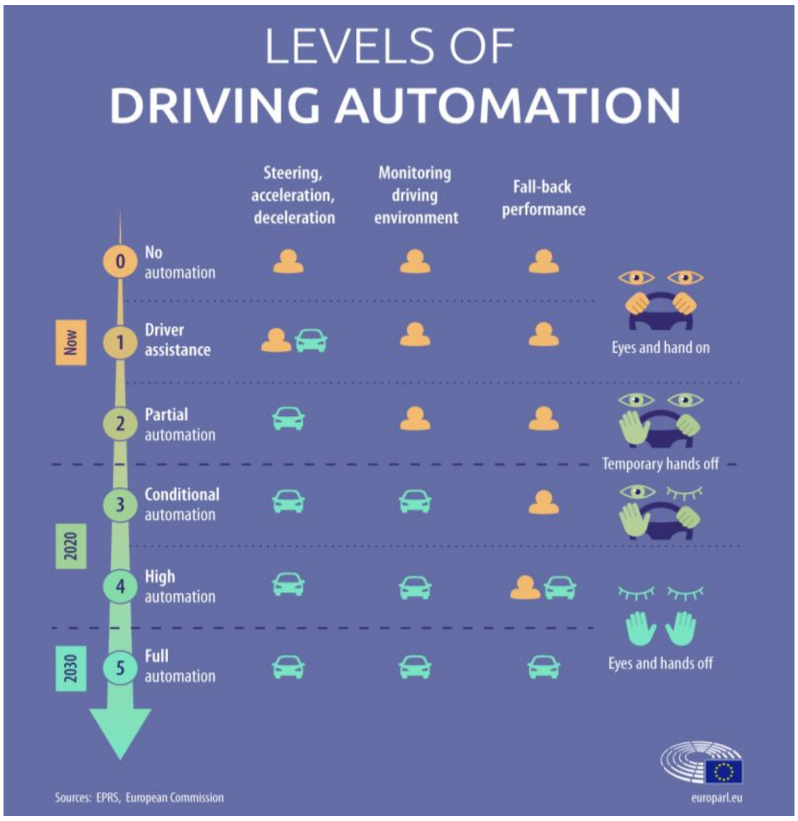


Figure 1: Levels of Driving Automation. Source: EPRS, European Commission [2]

The main aim of this thesis is to set a base on the topic of autonomous driving with the help of the open-source software CARLA in which we will be able to set up an environment and simulate the learning of an intelligent car.

1.2 Terms and concepts

The key concepts of this work are defined in the next section as the reader must be familiarised with the following concepts and understand them correctly as well as the topics related to autonomous driving that can generate more confusion.

1.2.1 Machine learning

Machine learning [3] is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns, and make decisions with minimal human intervention.

1.2.2 Reinforcement learning

Reinforcement learning [4] is a machine learning training method based on rewarding desired behaviours and/or punishing undesired ones. In general, a reinforcement learning agent is able to perceive and interpret its environment, take actions and learn through trial and error.

1.2.3 DQN

The Deep Q-Network algorithm [5] was developed by DeepMind in 2015. It could solve a wide range of Atari games (some to superhuman level) by combining reinforcement learning and deep neural networks at scale. The algorithm was developed by enhancing a classic RL algorithm called Q-Learning with deep neural networks and a technique called experience replay.

1.2.4 ANNs

Neural networks [6], also known as artificial neural networks (ANNs) are a subset of machine learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.

1.2.5 Autonomous Car

A self-driving car [7], also known as an autonomous vehicle (AV) or driverless car, is a car incorporating vehicular automation, that is, a ground vehicle that can sense its environment and moving safely with little or no human input

1.2.6 CARLA

CARLA [8] is the open-source simulator for autonomous driving research. It has been developed from the ground up to support development, training, and validation of autonomous driving systems. The simulation platform supports flexible specification of sensor suites, environmental conditions, full control of all static and dynamic actors, map generation and much more.

1.3 Problem to be resolved

The project consists of using the CARLA Simulator to obtain autonomous driving. It has been divided into two sections to facilitate and ensure the correct completion of the work. The first part consists of setting up the environment that we will be using for the execution of the learning process. This includes the prerequisites, software installation, library dependencies and any other requisite. The second part of the project is the proper learning of the car. To accomplish this a circuit will be defined with an origin, a destination, and a series of waypoints where the car must pass through. Once the waypoint system has been developed, we can start with the learning process of the car. Thanks to reinforcement learning, the car will perceive and interpret its environment, take actions, and learn through trial and error.

1.4 Stakeholders

This project has many involved parties, in a direct or indirect way, depending on the benefits and contribution they have with the project.

1.4.1 CARLA Community

One of the main beneficiaries will be the CARLA Community as most of the results and works obtained from this project will be easily reproduced and serve as feedback for them.

1.4.2 Director

My director, Gerard Escudero Bakx, is a direct stakeholder of my project because we will have an implication on the thesis as he guides me during the execution of it.

1.4.3 Myself

I am a direct stakeholder of this project. The grade of this project will decide if I will turn into a computer science engineer or not.

2. Justification

Most real-world reinforcement learning problems have an incredibly complicated state, environments, and actors. Therefore, any existing solution or study will be appreciated. That's why my director, Gerard Escudero Bakx, has provided me with two alumni theses to use in my favour. To obtain the best result for the project using and taking advantage of previous solutions will be mandatory. Adapting an existing study will help in achieving a better outcome for the thesis thanks to code reusability. Obviously, this will be a time-consuming decision due to the understanding and learning of a foreign project. Not using existing projects can be a major drawback since nothing ensures that we will achieve a significant advance in our study.

3. Project Scope

3.1 Objectives

This thesis has two main objectives. The first goal is to create and set up an environment for CARLA. This includes installing the prerequisites, the software used for the execution of the training and the libraries.

The second goal of my thesis is to try to recreate an autonomous driving simulation as close as possible to real life. Starting first with a simple scenario such as, the car must travel a route that simply follows a straight line and ending with the car travelling through a much more complex route.

3.2 Sub-Objectives

All the sub-objectives come from the second goal of the thesis.

3.2.1 Waypoints

Firstly, a circuit will be created where origin and destination points will be defined, and a series of waypoints will be established through which the car will have to pass. The distance between waypoints will be 1 meter.

3.2.2 Reinforcement Learning

Secondly, the car will use reinforcement learning to perceive, interpret its environment, take actions and learn through trial and error. This objective will be achieved with the correct implementation of a control algorithm and a reward system which the car will use to know when a good or bad choice has been made.

3.3 Requirements

The following requirements are needed to ensure the quality of the thesis

3.3.1 Functional requirements

The functional requirements [9] for this project are:

- Setting up a proper Python3 environment on Window
- A visual demonstration of the car's apprenticeship.
- Correct tuning of the hyper-parameters for the artificial neural network.

3.3.2 Non-functional Requirements

In general, a good use of programming language to obtain the following non-functional requirements: reusability, performance, and portability

3.4 Obstacles and Risks

Every project has different difficulties that must be analysed in advance to minimise their impact. Here are some of the risks that need to be aware of:

- Hyperparameters. Not tuning hyperparameters correctly is one of the biggest mistakes in hyperparameter optimization [10]. If they are not set explicitly on the model and, instead, model developer's defaults are used, they may be inappropriate.
- Computational power - Given the high computational requirements of neural network training, it is necessary for the computer running the learning to have a high performing CPU and GPU, as this may limit the development itself.
- Deadlines - Delivery deadlines must be achieved at all costs. Lack of experience may be a drawback as I'm not familiar with this type of work.
- Inexperience - As said before, inexperience can be a major disadvantage as learning and deciding the best way to develop this project will be a time-consuming task.

- Incompatibilities - Libraries incompatibilities are a big issue as some standard library interfaces may change during updates in ways that require different code than normal for example: Python3 code to achieve Python2/3 compatibility.

3.5 Changes in Project Scope

During the initial planning of the project the objectives where different. At first, we only had one main objective: to recreate an autonomous driving simulation as close as possible to real life. Then during the development of project, I realised that this main goal should be split into two as setting up the environment was a bigger task than previously planned. With this objective, we also added a risk that at first, we did not think about it, library incompatibilities.

4. Methodology and rigour

This section will justify which methodology has been chosen and the reasons for the decision, a description of the tools to be used to develop the project and how the objectives will be validated.

4.1 Agile Methodology

The work methodology that will be used in this project will be agile [11]. Agile is an iterative approach to project management that helps teams deliver their product. Instead of clustering everything on a single launch the agile methodology divides the work in several sprints in which teams plan, design, develop, test, deploy and review constantly. In this case, sprints will last two weeks, starting and finishing with a meeting with the project director.

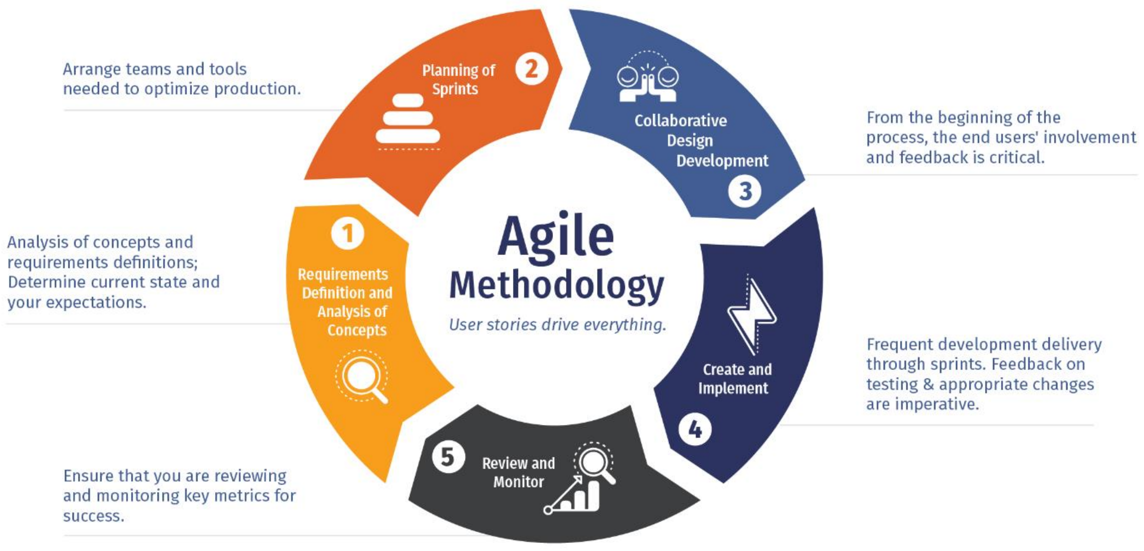

Arrange teams and tools needed to optimize production.
Planning of Sprints
2
Collaborative Design Development
3
From the beginning of the process, the end users' involvement and feedback is critical.
Analysis of concepts and requirements definitions; Determine current state and your expectations.
1
Requirements Definition and Analysis of Concepts
Agile Methodology
User stories drive everything.
Create and Implement
4
Frequent development delivery through sprints. Feedback on testing & appropriate changes are imperative.
5
Review and Monitor
Ensure that you are reviewing and monitoring key metrics for success.

Figure 2: Agile Methodology. Source: NVISIA, The Agile Process 101 [12]

The main reasons for choosing this methodology are:

- Project predictability - Constant meeting with the project's director will increase transparency making it easier to predict risk and plan for a smooth project execution.

- Continuous improvements - This methodology works by iterations which means that each sprint will be better than the previous one and mistakes done in the past will not be repeated.

- Better project quality - In agile project management, testing is always present in all sprints which implies that the overall quality of the final product is greater.

4.2 Tools and Validation

The tools that I'll be using for my project development will be the open-source simulator for autonomous driving research, CARLA Simulator, Anaconda [13] and a Virtual Machine for Windows with Python 3 in which I'll have an environment with all the libraries and dependencies needed to run de CARLA software. This environment will be set up with Anaconda.

Thanks to agile methodology, every two weeks meetings with the project's director will be arranged in which evaluation, objectives and even requirements will be discussed.

5. Task Definition

The project will need a total of 550 hours. The work-to-begin date is on February 21st, same day as the first day of GEP, and will be delivered on June 11th. I expect an average of 5 hours of work per day among 110 days which is the time difference between the two dates.

5.1 Tasks Description

In this section I will identify and describe all tasks that are involved in this study.

T1 - Project Management

This task includes the context and scope task, project planning task, budget and sustainability task, final project definition task and the meetings task.

T1.1 - Context and Scope

In this task I will contextualise the thesis and describe its scope. I estimate that this task will consume a total of 20 hours.

T1.2 - Project Planning

In this task I will plan the start and end date, the hours per day and a brief description of all the tasks of this project. I estimate that this task will consume a total of 10 hours.

T1.3 - Budget and Sustainability

In this task I will draw up the budget of the project and the initial part of the sustainability report. I estimate that this task will consume a total of 15 hours.

T1.4 - Final Project Definition

In this task I will integrate the previous tasks (T1.1, T1.2 and T1.3) in a final document. I estimate that this task will consume a total of 15 hours.

T1.5 - Meetings

This task consists of carrying out meetings with the director of the project throughout the weeks. I estimate that this task will consume a total of 20 hours.

T2 - Research

This part includes all the research tasks needed to start the implementation of the project.

T2.1 - Get familiarised with ANNs

This task consists of learning and getting familiar with the concept of ANNs. I estimate that it will take a total of 25 hours to complete.

T2.2 - Get familiarised with PPO

This task consists of learning and getting familiar with the concept of PPO. I estimate that it will consume a total of 25 hours.

T2.3 - Learn about CARLA simulator

This task consists of learning and getting familiar with the CARLA simulator environment. I estimate that it will take a total of 40 hours to complete.

T2.4 - Learn KERAS and TensorFlow

This task consists of learning and getting familiar with the KERAS and TensorFlow. I estimate that it will take a total of 20 hours to complete.

T3Pre - Setting up

This part includes all the tasks that consist of preparing the environment for the CARLA simulator.

T3Pre.1 - Installing prerequisites

This part consists of installing all the necessary prerequisites to be able to start with the installation of the software. I estimate that this task will take a total of 10 hours.

T3Pre.2 Installing software

This part consists of installing all the necessary prerequisites to be able to start with the installation of the libraries. I estimate that this task will take a total of 20 hours.

T3Pre.3 Installing libraries

This part consists of installing all the necessary libraries. Once it’s finished, I will be able to start with the programming part. I estimate that this task will take a total of 20 hours.

T3 - Programming part

This part includes all the tasks that consist of programming in Python3 or in the CARLA simulator.

T3.1 - Designing map levels

For this task I will be designing five different maps in which the autonomous car will drive and learn. This task will cost 20 hours.

T3.2 - Implementation of waypoints

This task consists of implementing the waypoints system that the car will use to move around the map. This task will consume 20 hours.

T3.3 - Implementation of reward system

This task consists of implementing the reward system that the car and the ANN will use to learn what is “right” or “wrong”. This task will cost 40 hours.

T3.4 - Implementation of RL in car

This task consists of implementing the reinforcement learning algorithm with the proximal policy optimization and including the waypoint and reward system. This task will cost 50 hours of work.

T4 - Experiments, Analysis and Conclusion

This task consists of the testing part of the project. Which includes, tuning the hyperparameters of the ANN, testing the car in different map levels, testing the car with different map difficulties, and collecting all the data from the experiments.

T4.1 - Tuning hyperparameters

During experiments, I will be fiddling with the hyperparameters in order to approach the best solution. I estimate that this task will consume a total of 15 hours.

T4.2 - Testing in different map levels

During the training of the car, I will be changing the maps to obtain the best outcome from the ANN’s. I expect that this task will consume 25 hours of work.

T4.3 - Testing with increased level difficulty

During the training of the car, I will also be changing the map difficulty. I'll be adding traffic lights, other cars, pedestrians, and even realistic weather. This task will consume a total of 30 hours.

T4.4 - Collecting data

During all the experiments tasks (T4.1, T4.2, T4.3), I'll be collecting data obtained by the autonomous car. I estimate that this task will take 30 hours of work.

T5 - Project Documentation

When I have finished task T4 (experiments, analysis, and conclusion), I will be able to start with the project documentation. I'll be using Google Docs and this task will consume 50 hours.

T6 - Bachelor Thesis Defence Preparation

In this task I'll be preparing and practising the defence of my bachelor thesis. Obviously, this task has a dependency with T5, and I will take a total of 30 hours of work.

5.2 Resources

In this section we will be mentioning the human and material resources used in this research project.

5.2.1 Human Resources

The main human resource of the thesis is:

- The researcher, Enrique Martinez Martel
- Director of the project, Gerard Escudero Bakx
- GEP Tutor, Paola Lorenza Pinto

5.2.2 Material Resources

The material resources are the following:

- **PC -** The computer used to carry out the project is a MSI GE75 Raider 8SE with an Intel(R) Core (TM) i7-8750H CPU, 16 GB of RAM and a NVIDIA RTX 2060 GPU.

- **ATENEA -** The UPC's virtual learning environment.

- **Google Meet -**The video-communication service developed by Google used for the with the director of the project.

- **CARLA -** The open-source simulator for autonomous driving research used to carry out important tasks.

- **Python3 -** Programming language used for this project.

- **Anaconda** - Anaconda is a distribution of the Python and R programming languages for scientific computing that aims to simplify package management and deployment.

- **Visual Studio Code** - VS Code, is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

5.3 Changes in Task Definition

During the initial planning of the project the objectives where different. Therefore, there was a different task planification. Previously we didn’t have any T3Pre tasks as they were created with the introduction of the new objective: setting up an environment for CARLA. With this changes, the total time increased from 500 hours to 550 and the delivery date was postponed from June 1st to June 11th.

6. Estimates and the Gantt

In this section, we have an overview of the workload estimation in Table 1 and the Gantt diagram in Figure 3.

6.1 Workload estimation

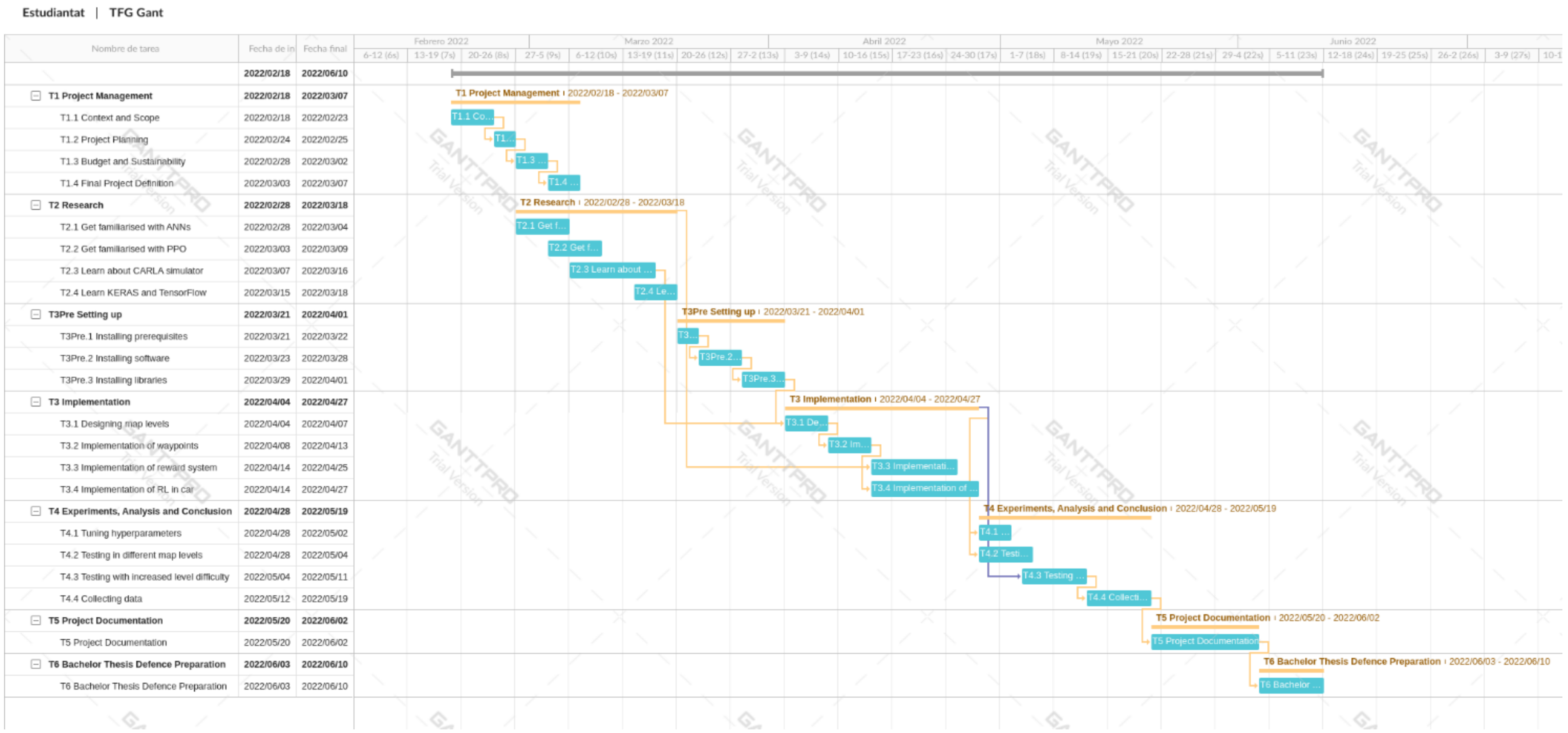
The following table summarises the previously explained tasks. Each task has an ID for identification, a name, time estimated for completion, dependencies and the resources needed.

ID	Name	Time(h)	Dependencies	Resources
T1	Project Management	80		
T1.1	Context and Scope	20		PC, Atenea
T1.2	Project Planning	10	T1.1	PC, Atenea
T1.3	Budget and Sustainability	15	T1.1, T1.2	PC, Atenea
T1.4	Final Project Definition	15	T1.1, T1.2, T1.3	PC, Atenea
T1.5	Meetings	20		PC, Google Meet
T2	Research	110		
T2.1	Get familiarised with ANNs	25		PC
T2.2	Get familiarised with PPO	25		PC
T2.3	Learn about CARLA simulator	40		PC, CARLA, Python3
T2.4	Learn KERAS and TensorFlow	20		PC
T3Pre	Setting up	50		
T3Pre.1	Installing prerequisites	10		PC
T3Pre.2	Installing software	20	T3Pre.1	PC, CARLA, Python3, Anaconda
T3Pre.3	Installing libraries	20	T3Pre.1, T3Pre.2	PC, CARLA, Python3, Anaconda
T3	Implementation	130		
T3.1	Designing map levels	20	T2.3, T3Pre	PC, CARLA, Python3, Anaconda, VSC

T3.2	Implementation of waypoints	20	T2.3, T3.1	PC, CARLA, Python3, Anaconda, VSC
T3.3	Implementation of reward system	40	T2	PC, CARLA, Python3, Anaconda, VSC
T3.4	Implementation of RL in car	50	T2, T3.2, T3.3	PC, CARLA, Python3, Anaconda, VSC
T4	Experiments, Analysis and Conclusion	100		
T4.1	Tuning hyperparameters	15	T3	PC, CARLA, Python3, Anaconda, VSC
T4.2	Testing in different map levels	25	T3	PC, CARLA
T4.3	Testing with increased level difficulty	30	T3	PC, CARLA
T4.4	Collecting data	30	T3	PC, Google Docs
T5	Project Documentation	50	T4	PC, Google Docs
T6	Bachelor Thesis Defence Preparation	30	T5	PC
Total		550		

Table 1: Summary of the tasks. Source: Own compilation

6.2 Gantt Chart


Estudiantat | TFG Gant

Nombre de tarea	Fecha de in	Fecha final
	2022/02/18	**2022/06/10**
T1 Project Management	**2022/02/18**	**2022/03/07**
T1.1 Context and Scope	2022/02/18	2022/02/23
T1.2 Project Planning	2022/02/24	2022/02/25
T1.3 Budget and Sustainability	2022/02/28	2022/03/02
T1.4 Final Project Definition	2022/03/03	2022/03/07
T2 Research	**2022/02/28**	**2022/03/18**
T2.1 Get familiarised with ANNs	2022/02/28	2022/03/04
T2.2 Get familiarised with PPO	2022/03/03	2022/03/09
T2.3 Learn about CARLA simulator	2022/03/07	2022/03/16
T2.4 Learn KERAS and TensorFlow	2022/03/15	2022/03/18
T3Pre Setting up	**2022/03/21**	**2022/04/01**
T3Pre.1 Installing prerequisites	2022/03/21	2022/03/22
T3Pre.2 Installing software	2022/03/23	2022/03/28
T3Pre.3 Installing libraries	2022/03/29	2022/04/01
T3 Implementation	**2022/04/04**	**2022/04/27**
T3.1 Designing map levels	2022/04/04	2022/04/07
T3.2 Implementation of waypoints	2022/04/08	2022/04/13
T3.3 Implementation of reward system	2022/04/14	2022/04/25
T3.4 Implementation of RL in car	2022/04/14	2022/04/27
T4 Experiments, Analysis and Conclusion	**2022/04/28**	**2022/05/19**
T4.1 Tuning hyperparameters	2022/04/28	2022/05/02
T4.2 Testing in different map levels	2022/04/28	2022/05/04
T4.3 Testing with increased level difficulty	2022/05/04	2022/05/11
T4.4 Collecting data	2022/05/12	2022/05/19
T5 Project Documentation	**2022/05/20**	**2022/06/02**
T5 Project Documentation	2022/05/20	2022/06/02
T6 Bachelor Thesis Defence Preparation	**2022/06/03**	**2022/06/10**
T6 Bachelor Thesis Defence Preparation	2022/06/03	2022/06/10

Febrero 2022
Marzo 2022
Abril 2022
Mayo 2022
Junio 2022
6-12 (6s)
13-19 (7s)
20-26 (8s)
27-5 (9s)
6-12 (10s)
13-19 (11s)
20-26 (12s)
27-2 (13s)
3-9 (14s)
10-16 (15s)
17-23 (16s)
24-30 (17s)
1-7 (18s)
8-14 (19s)
15-21 (20s)
22-28 (21s)
29-4 (22s)
5-11 (23s)
12-18 (24s)
19-25 (25s)
26-2 (26s)
3-9 (27s)
T1 Project Management | 2022/02/18 - 2022/03/07
T2 Research | 2022/02/28 - 2022/03/18
T3Pre Setting up | 2022/03/21 - 2022/04/01
T3 Implementation | 2022/04/04 - 2022/04/27
T4 Experiments, Analysis and Conclusion | 2022/04/28 - 2022/05/19
T5 Project Documentation | 2022/05/20 - 2022/06/02
T6 Bachelor Thesis Defence Preparation | 2022/06/03 - 2022/06/10

Figure 3: Gantt Chart. Source: Own compilation

6.3 Changes in Estimates and Gantt Chart

The workload estimation changed from the first planification because the task T3Pre was added after. Therefore, the workload estimation and the Gantt Chart changed (see Appendix A).

7. Risk Management

It is important to predict the risks and obstacles that may emerge during the project's development. In this section you will find a description of the problems mentioned in the first deliverable and the alternative plans to overcome them.

7.1 Hyperparameter tuning

Not tuning hyperparameters correctly is one of the biggest mistakes in hyperparameter optimization. If they are not set explicitly on the model and, instead, model developer's defaults are used, they may be inappropriate.

- **Impact:** Medium
- **Alternative plan:** To solve the problem, I could adapt the project planning and add a new task that would focus on hyperparameter tuning. I estimate that this new task will take 15 hours.

7.2 Computational power

Given the high computational requirements of neural network training, it is necessary a high performing CPU and GPU, as this may limit the development itself.

- **Impact:** Low
- **Alternative plan:** To solve the problem, I will plan to use my resources wisely. I will try to train my neural network at night when I'm not working on my laptop. This alternative plan can bring problems such as: not choosing the right learning rate, not choosing the appropriate number of epochs or iterations, or not knowing when to stop the training. Nevertheless, it's better than not doing anything. I estimate that this task will cost 10 hours.

7.3 Deadline

Due to academic deadlines, the project should not be extended more than the initial time planned.

- **Impact:** High
- **Alternative plan:** To solve the problem, I could recalculate every week in the project meetings the hours of my time planning. Therefore, I could approach a more realistic and updated time schedule. I estimate that this task will cost 15 hours.

7.4 Inexperience

Inexperience can be a major disadvantage as I have never used the CARLA simulator environment, worked with the proximal policy optimization (PPO) or even the neural networks before. Thus, this problem should be dealt with carefully.

- **Impact:** Medium
- **Alternative plan:** To solve the problem, I could adapt the project planning and extend time on the tasks that I find more difficult. I estimate that this extension would be never more than 10 hours for each task.

7.5 Incompatibilities

Inexperience can be a major disadvantage as I have never used the CARLA simulator environment, worked with the proximal policy optimization (PPO) or even the neural networks before. Thus, this problem should be dealt with carefully.

- **Impact:** High
- **Alternative plan:** To solve the problem, I could extend the duration for the setting up of the environment task. I estimate that this task will cost 20 more hours.

8. Budget

In this section I will talk about the project's budget. It will include the different types of costs associated with the personnel's costs by activity, the hardware and software used and the workspace. In addition, to overcome the obstacles that may arise during the development, we have created a contingency plan with an estimated budget and an estimation cost for the incidents that can occur.

8.1 Identification of costs and cost estimates

8.1.1 Personnel cost per activity

In this project there are 4 types of different personnel, and they all have different costs per hour. Firstly, we have the project manager, who is responsible for the planning and execution of the project. Secondly, we have the researcher, who will oversee collecting information about the project and technologies. Thirdly, we have the programmer who will be doing all the implementation of the project. Finally, we have the tester who will be responsible for the testing and reviewing the work done by the programmer.

Role	**Cost per hour without SS**	**Cost per hour with SS**
Project Manager	23,07€/hour	30 €/hour
Researcher	15,38 €/hour	20 €/hour
Programmer	12,30 €/hour	16 €/hour
Tester	12,30 €/hour	16 €/hour

Table 2: Personnel costs. Source: Hays labour market guide [14]

ID	Activity	Cost (€)	Comments
T1.1	Context and Scope	600	20 hours, Project Manager
T1.2	Project Planning	300	10 hours, Project Manager
T1.3	Budget and Sustainability	450	15 hours, Project Manager
T1.4	Final Project Definition	450	15 hours, Project Manager
T1.5	Meetings	600	20 hours, Project Manager

T2.1	Get familiarised with ANNs	500	25 hours, Researcher
T2.2	Get familiarised with PPO	500	25 hours, Researcher
T2.3	Learn about CARLA simulator	360	40 hours, Researcher
T2.4	Learn KERAS and TensorFlow	400	20 hours, Researcher
T3Pre.1	Installing prerequisites	160	10 hours, Programmer
T3Pre.2	Installing software	320	20 hours, Programmer
T3Pre.3	Installing libraries	320	20 hours, Programmer
T3.1	Designing map levels	320	20 hours, Programmer
T3.2	Implementation of waypoints	320	20 hours, Programmer
T3.3	Implementation of reward system	640	40 hours, Programmer
T3.4	Implementation of RL in car	800	50 hours, Programmer
T4.1	Tuning hyperparameters	240	15 hours, Tester
T4.2	Testing in different map levels	400	25 hours, Tester
T4.3	Testing with increased level difficulty	480	30 hours, Tester
T4.4	Collecting data	600	30 hours, Researcher
T5	Project Documentation	1500	50 hours, Project Manager
T6	Bachelor Thesis Defence Preparation	900	30 hours, Project Manager
	TOTAL PERSONNEL COST BY ACTIVITY	**11160**	**550 hours, All Roles**

Table 3: Total Personnel Cost. Source: Own compilation

8.1.2 Generic costs

Hardware and software

In this project, we only have amortisations for hardware because all the software used is open source or the UPC has given us a licence to use it. The resources will be used for a total of 550 hours. The amortisation costs are shown below:

$$Amortisation\ (€) = \ ResourcePrice\ \chi\ \frac{1}{4years}\ \chi\ \frac{1}{100days}\ \chi\ \frac{1}{5hours}\ \chi\ HoursUsed$$

<u>**Hardware**</u>	<u>**Price**</u>	<u>**Time used(h)**</u>	<u>**Amortisation (€)**</u>
Laptop	1650	550	453.75
Display	110	550	30.25

Hardware	**Price**	**Time used(h)**	**Amortisation (€)**
Laptop	1650	550	453.75
Keyboard	50	550	13.75
Mouse	30	550	8.25
Microphone	50	20	0.5
		TOTAL	**506,5**

Table 4: Hardware and Software Total Cost. Source: Own compilation

Workspace
In this section we will calculate the total cost for the workspace. This includes the electricity, furniture, and internet. For the **furniture** we used the following formula and obtained a total of 300*(¼)*(1/100) *(⅕)*550 = 82.5€.

$$TotalCostOfFurniture\ \chi\ \frac{1}{4years}\chi\ \frac{1}{100days}\chi\ \frac{1}{5hours}\chi\ HoursUsed$$

Considering that the price of the kWh in Spain is 0,3544 €/kWh [15] and the total hours of consumption will be 550, we will calculate the total cost of **electricity** and obtain 34,55€.

Hardware	**Power (W)**	**Time used (h)**	**Consumption (kWh)**	**Price (€)**
Laptop	150	550	75	26,58
Display	45	550	22,5	7,97
			TOTAL	**34,55**

Table 5: Hardware and Software Total Cost. Source: Own compilation

The **internet** costs 25€ per month. Considering that the project lasts 4 months (110 days) and that the working hours per day are 5 the internet cost is 4 months * (30€/month) * (5h/24h) = 25€.

In conclusion, we have a total generic cost of 648,55€ and if we sum the total personnel cost by activity, we obtain a total of 648,55€+11160€= 11808,55€.

8.1.3 Contingencies

During the project's development, potentially negative events can occur. Therefore, it is always necessary to prepare a budget to reduce the impact as much as possible. In this case we have a contingency margin of 15%, 1771,28€.

8.1.4 Incidental costs

We also need to consider the possible cost of applying alternative plans in case of unforeseen events. These plans have been discussed previously and the total cost of them are calculated in the following table.

Risks	Estimated hours (h)	Risk (%)	Cost (€)
Hyperparameter tuning	15	20	90
Computational power	10	10	30
Deadline	15	15	67,5
Inexperience	10	20	60
Incompatibilities	20	20	60
		TOTAL	**337,5**

Table 6: Incidental Costs. Source: Own compilation

ID	Activity	Cost (€)	Comments
T1.1	Context and Scope	600	20 hours, Project Manager
T1.2	Project Planning	300	10 hours, Project Manager
T1.3	Budget and Sustainability	450	15 hours, Project Manager
T1.4	Final Project Definition	450	15 hours, Project Manager
T1.5	Meetings	600	20 hours, Project Manager
T2.1	Get familiarised with ANNs	500	25 hours, Researcher
T2.2	Get familiarised with PPO	500	25 hours, Researcher
T2.3	Learn about CARLA simulator	360	40 hours, Researcher
T2.4	Learn KERAS and TensorFlow	400	20 hours, Researcher
T3Pre.1	Installing prerequisites	160	10 hours, Programmer
T3Pre.2	Installing software	320	20 hours, Programmer
T3Pre.3	Installing libraries	320	20 hours, Programmer
T3.1	Designing map levels	320	20 hours, Programmer
T3.2	Implementation of waypoints	320	20 hours, Programmer
T3.3	Implementation of reward system	640	40 hours, Programmer
T3.4	Implementation of RL in car	800	50 hours, Programmer
T4.1	Tuning hyperparameters	240	15 hours, Tester
T4.2	Testing in different map levels	400	25 hours, Tester
T4.3	Testing with increased level difficulty	480	30 hours, Tester

T4.4	Collecting data	600	30 hours, Researcher
T5	Project Documentation	1500	50 hours, Project Manager
T6	Bachelor Thesis Defence Preparation	900	30 hours, Project Manager
	Total CPA	11160	Total personnel costs by activity (Gantt activities)
	Hardware		
H1	Laptop	453,75	Amortisation
H2	Display	30,25	Amortisation
H3	Keyboard	13,75	Amortisation
H4	Mouse	8,25	Amortisation
H5	Microphone	0,5	Amortisation
	Software		
S1	CARLA Software	0	Free to use
S2	VisualStudio	0	Free to use
S3	Python3	0	Free to use
S4	Microsoft Office	0	UPC licence
S5	Google Meet	0	Free to use
S6	GanttProject	0	Free to use
	Workspace		
W1	Electricity	34,55	
W2	Furniture	82,5	
W3	Internet	25	
	Total GC	649,55	Total Costs imputed generically (not detailed by activity)
	Total CPA + Total GC	11808,05	Total Costs
	Contingency	1771,29	Contingency margin of 15%
	Total CPA + Total GC + Contingency	13579,83	
	Risks		
R1	Hyperparameter tuning	90	15 hours, risk = 20%, cost per hour = 30€
R2	Computational power	30	10 hours, risk = 10%, cost per hour = 30€
R3	Deadline	67,5	15 hours, risk 15%, cost per hour = 30€
R4	Inexperience	60	10 hours, risk 20%, cost per hour = 30€
R5	Incompatibilities	90	20 hours, risk 20%, cost per hour = 30€
	Total incidentals	337,5	
	Total	**13917,33**	**Total cost: CPA+CG+Contingency+Incidentals**

Table 7: Total Project Cost. Source: Own compilation

8.2 Management control

In this section, we will describe the procedures and methods used to control the budget. Also, we will define the control indicators that will help us supervise the cost deviations during the development of the project. Control mechanisms are the difference between the real and estimated resource consumptions, and they are used to measure and avoid deviations.

This is the formula used to obtain the personnel cost deviation:

For the hardware and software deviation:
*Hardware Software Deviation= (EstimatedCostPerHours -RealCostPerHours)*TotalHoursConsumed*

For the workspace deviation:
*Workspace Cost Deviation = (EstimatedCostPerHours - RealCostPerHours) * TotalHoursConsumed*

For the incidental cost deviation:
*Incidental Cost Deviation =(EstimatedIncidentalHours - RealIncidentalHours)*TotalIncidentalHours*

Finally, we need to sum up all if we want to know the general costs deviation:

Costs Deviation = Personnel Cost Deviation + Hardware Software Deviation
+ Workspace Cost Deviation + Incidental Cost Deviation

All these indicators will help us know where the deviations in the budget are.

9. Sustainability report

9.1 Self-assessment

After reading the first question from the pool I realised that my knowledge of sustainability was poor. Therefore, I started by looking for the definition of the word. Sustainability is a broad policy concept in the global public discourse and is often conceived of in terms of three pillars: environmental, economic, and social. The original semantic meaning of “sustainability” and “to sustain” refers to the ability to continue over a long period of time. Once I had the correct definition of sustainability, I started the poll.

At first, I could not give an answer to many of the questions of the pool because I simply never thought about it. I used to think that the basics of sustainability was to reduce the CO2 emissions, but I was totally wrong. A sustainability report should consider how I will approach sustainable development, what are the social, economic, and environmental impacts of the product, the health, safety, and social justice implications of my product, how to measure it (ISO 26000, 2014/95/EU Directive), what are the methods and tools for estimating the economic feasibility of a project and the deontological principles of my product and the ethical principles of sustainability.

Throughout the questionnaire, I concluded that there are many things we overlook in our day-to-day life. All businesses need to follow a sustainable report to minimise their footprint, innovate around the life cycle of a product and be transparent. This brings several benefits such as: improved brand image, increase productivity, reduce costs, attract employees and investors, reduce waste, and even make shareholders happy.

To summarise the above, this self-assessment has made me understand the importance of sustainability and the little I knew about it. A project, idea or product that does not take this into account will negatively affect our society, economics and environment and probably will be doomed to failure.

9.2 Economic

Regarding PPP, have you estimated the environmental impact of undertaking the project? Have you considered how to minimise the impact, for example by reusing resources?

The cost of undertaking the project, human and material resources, estimation can be found in Section 8 of the document.

Regarding useful life, how is the problem that you wish to address resolved currently (state of the art)? In what ways will your solution environmentally improve existing solutions?

Studies and research of autonomous driving are very expensive as knowledge and machinery are needed to carry them out. Therefore, for my solution I'll be reusing and referring to another existing thesis to reduce costs.

9.3 Environmental

Regarding PPP, have you estimated the cost of undertaking the project (human and material resources)?

My main tool for the project will be my own PC and the estimated environmental impact will depend entirely on the energy consumption of it. Minimising this will be difficult as the learning of ANNs is very unpredictable.

Regarding useful life, how is the problem that you wish to address resolved currently (state of the art)? In what ways will your solution economically improve existing solutions?

In my case, an environmental improvement from existing solutions will be difficult to obtain because other projects use more efficient and specific machines for RL. Nevertheless, my tool for this project will be my PC and the resource used in it will be the electricity to power it up. To obtain a better environmentally friendly solution I will need to be using an energy efficient PC and this could be achieved with the new GPUs as I cannot afford to use other specific tools such as supercomputers.

9.4 Social

Regarding PPP, what do you think undertaking the project has contributed to you personally?

This project will help me expand my understanding of what I am capable of, deepen my abilities to carry out a study and develop my skills in making connections between ideas. Also, this project will help me to introduce into the autonomous industry, particularly, in the pioneering topic of driverless cars.

Regarding useful life, how is the problem that you wish to address resolved currently (state of the art)? In what ways will your solution socially improve (quality of life) existing? Is there a real need for the project?

The main purpose of this thesis is to use CARLA for autonomous driving. As we all know, it is a new technology, so it is very difficult to find other theses, studies, or even small projects. That is why I hope my thesis promotes the development of other autonomous car projects. In other words, carrying out this study and making it accessible for others will encourage people to keep on working in this new field.

10. Environment Setup

10.1 Prerequisites

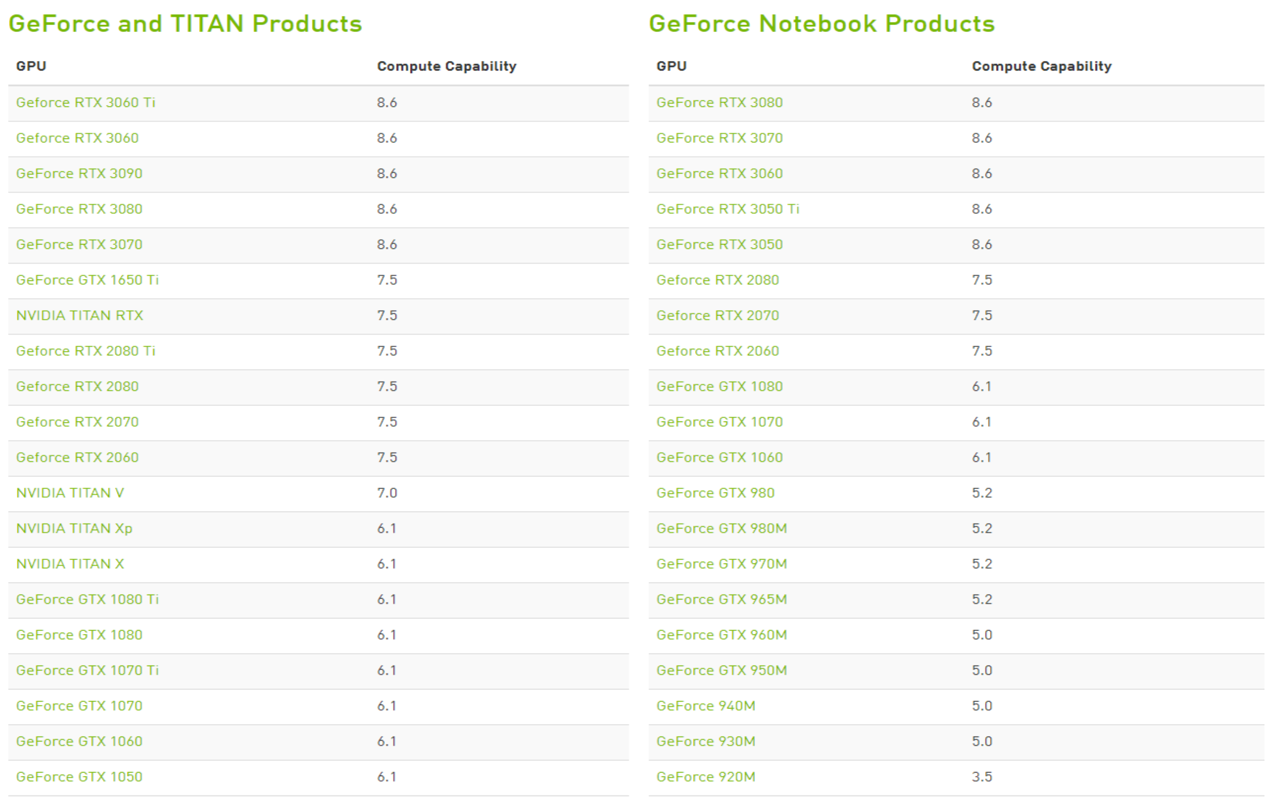
The following sections will specify all the requirements that should be fulfilled before installing CARLA. In this thesis the installation will be specifically for Windows the reason for this will be discussed later.

10.1.1 Operative System

CARLA is built for Windows and Linux systems but in my case, I'll be using Windows 10 specifically Microsoft Windows 10 Pro. This is due to some issues with the NVIDIA drivers on Linux. Therefore, to avoid these problems Windows was the winning option. Only a Windows 7 or higher (64-bit) will work for CARLA and if Linux was chosen a version of 64-bit Ubuntu 16.04 or higher will be necessary for the correction execution of CARLA.

10.1.2 GPU

CARLA aims for realistic simulations, so the server needs at least a 6 GB GPU although CARLA developers recommend 8 GB. A dedicated GPU is also highly recommended for machine learning because they can perform multiple simultaneous computations which enables the distribution of training processes and can significantly speed machine learning operations. In this case, the GPU should have a CUDA® architecture of 3.5, 5.0, 6.0, 7.0, 7.5, 8.0 or higher to benefit from this advantage. Figure 4 shows a list of the compatible GPUs.



GeForce and TITAN Products

GPU	Compute Capability
Geforce RTX 3060 Ti	8.6
Geforce RTX 3060	8.6
GeForce RTX 3090	8.6
GeForce RTX 3080	8.6
GeForce RTX 3070	8.6
GeForce GTX 1650 Ti	7.5
NVIDIA TITAN RTX	7.5
Geforce RTX 2080 Ti	7.5
Geforce RTX 2080	7.5
Geforce RTX 2070	7.5
Geforce RTX 2060	7.5
NVIDIA TITAN V	7.0
NVIDIA TITAN Xp	6.1
NVIDIA TITAN X	6.1
GeForce GTX 1080 Ti	6.1
GeForce GTX 1080	6.1
GeForce GTX 1070 Ti	6.1
GeForce GTX 1070	6.1
GeForce GTX 1060	6.1
GeForce GTX 1050	6.1

GeForce Notebook Products

GPU	Compute Capability
GeForce RTX 3080	8.6
GeForce RTX 3070	8.6
GeForce RTX 3060	8.6
GeForce RTX 3050 Ti	8.6
GeForce RTX 3050	8.6
Geforce RTX 2080	7.5
Geforce RTX 2070	7.5
Geforce RTX 2060	7.5
GeForce GTX 1080	6.1
GeForce GTX 1070	6.1
GeForce GTX 1060	6.1
GeForce GTX 980	5.2
GeForce GTX 980M	5.2
GeForce GTX 970M	5.2
GeForce GTX 965M	5.2
GeForce GTX 960M	5.0
GeForce GTX 950M	5.0
GeForce 940M	5.0
GeForce 930M	5.0
GeForce 920M	3.5

Figure 4: CUDA-Enabled GeForce and TITAN Products List. Source: NVIDIA website [16]

10.1.3 Disk Space

About 25GB in total will be required. Currently on CARLA 0.9.5, it needs a total amount of 5.9 GB of space. Depending on the version for CARLA and the addons you would like to use this number will change. For example, CARLA's 0.9.13 latest version with the AdditionalMaps addon [16] consumes a total amount of space of 8GB. All the remaining space will be occupied with the software dependencies.

10.1.4 System Overview

These are the development computer specs used during all the development and study of this thesis:

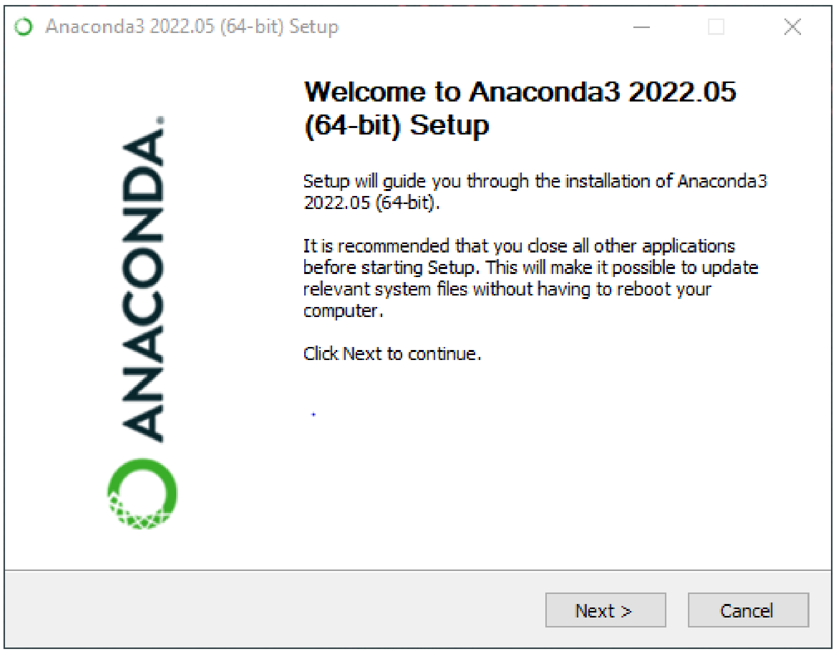
- Operative system: Microsoft Windows 10 Pro, 64-bit (21H2 Version)
- CPU: Intel(R) Core (TM) i7-8750H CPU @ 2.20GHz   2.21 GHz
- GPU: NVIDIA GeForce RTX 2060 (Laptop)
- RAM: 16,0 GB (15,8 GB available)
- Disk space: 250GB SDD + 1TB HDD

10.2 Environment Software

In this section will set and install all the software that will be used during the development of the thesis.

10.2.1 Anaconda

The first thing to do is to install Anaconda. This software is a distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. In this case we will be using the latest version of Anaconda for Windows.



Welcome to Anaconda3 2022.05 (64-bit) Setup

Setup will guide you through the installation of Anaconda3 2022.05 (64-bit).

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

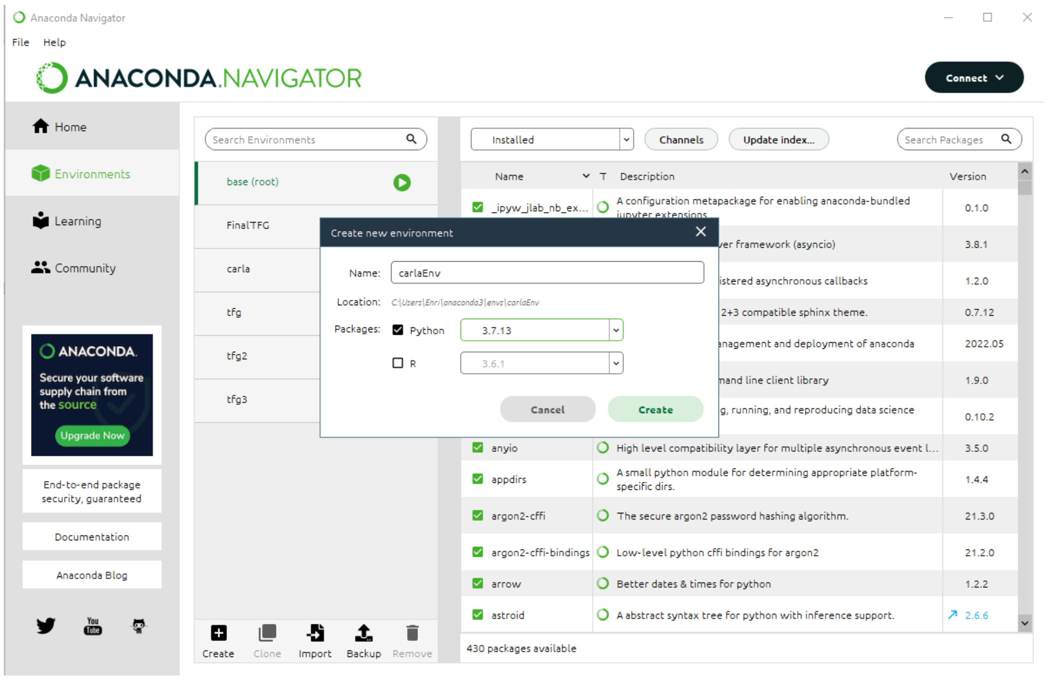
Click Next to continue.

Figure 5: Anaconda Installer. Source: Own compilation.

For more information about the chosen installation refer the appendix.

10.2.2 Python 3.7

Python is the main scripting language in this study. CARLA supports Python 2.7 and Python 3 on Linux, and Python 3 on Windows. To correctly run CARLA, we need Python 3.7, but the Anaconda's latest version has installed natively Python 3.9. This is the first issue that we can find in this thesis because this version has conflicts with many libraries such as Keras and Tensorflow. Therefore, we will need to make a new environment that will solve the issue. Figure 6 shows clearly how to create it in Anaconda.



Create new environment

Name: carlaEnv

Location: C:\Users\Enri\anaconda3\envs\carlaEnv

Packages: [✓ Python] 3.7.13 ☐ R 3.6.1

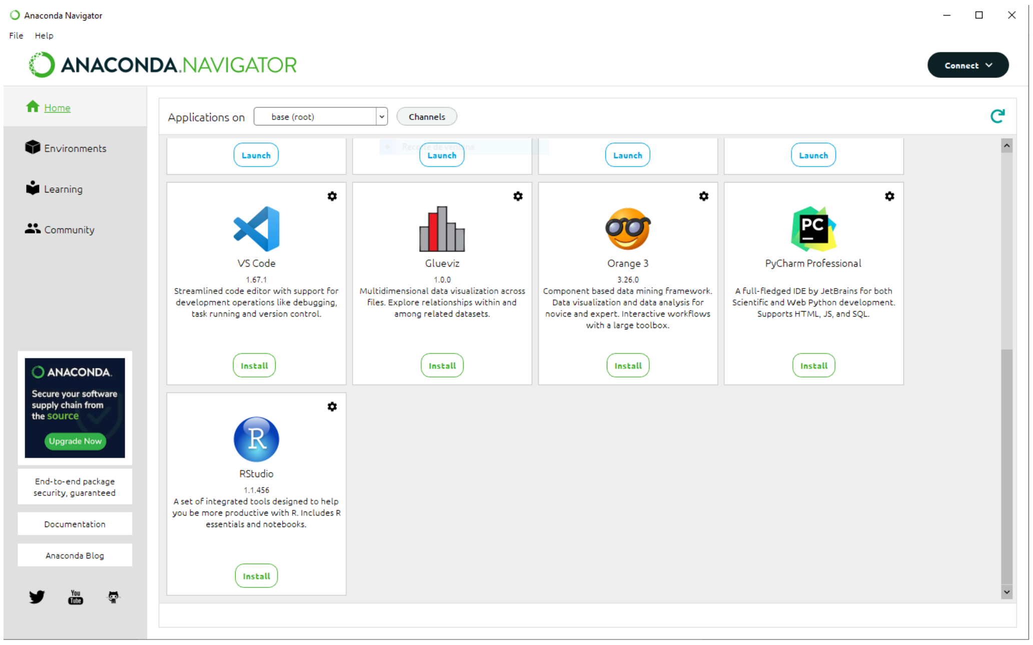
	Name	T	Description	Version
✓	_ipyw_jlab_nb_ex...		A configuration metapackage for enabling anaconda-bundled jupyter extensions	0.1.0
			ver framework (asyncio)	3.8.1
			istered asynchronous callbacks	1.2.0
			2+3 compatible sphinx theme.	0.7.12
			anagement and deployment of anaconda	2022.05
			mand line client library	1.9.0
			g, running, and reproducing data science	0.10.2
✓	anyio		High level compatibility layer for multiple asynchronous event l...	3.5.0
✓	appdirs		A small python module for determining appropriate platform-specific dirs.	1.4.4
✓	argon2-cffi		The secure argon2 password hashing algorithm.	21.3.0
✓	argon2-cffi-bindings		Low-level python cffi bindings for argon2	21.2.0
✓	arrow		Better dates & times for python	1.2.2
✓	astroid		A abstract syntax tree for python with inference support.	2.6.6

Figure 6: Anaconda Navigator. Source: Own compilation

We first select the Environments tab and click on the Create option. Then we name the environment, in this case its "carlaEnv" and we select the Python Package with the 3.7.13 version.

10.2.3 Visual Studio Code

Visual Studio Code [17] is a source-code editor made by Microsoft for Windows, Linux and macOS. It has features such as support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. In this thesis Visual Studio Code is a very useful tool as it will simply and make easier all the code development. To install the software, we will use the Anaconda Navigator. We will search for Visual Studio Code in the Anaconda Home page and simple select install as shown in Figure 7.



VS Code
1.67.1
Streamlined code editor with support for development operations like debugging, task running and version control.

Glueviz
1.0.0
Multidimensional data visualization across files. Explore relationships within and among related datasets.

Orange 3
3.26.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

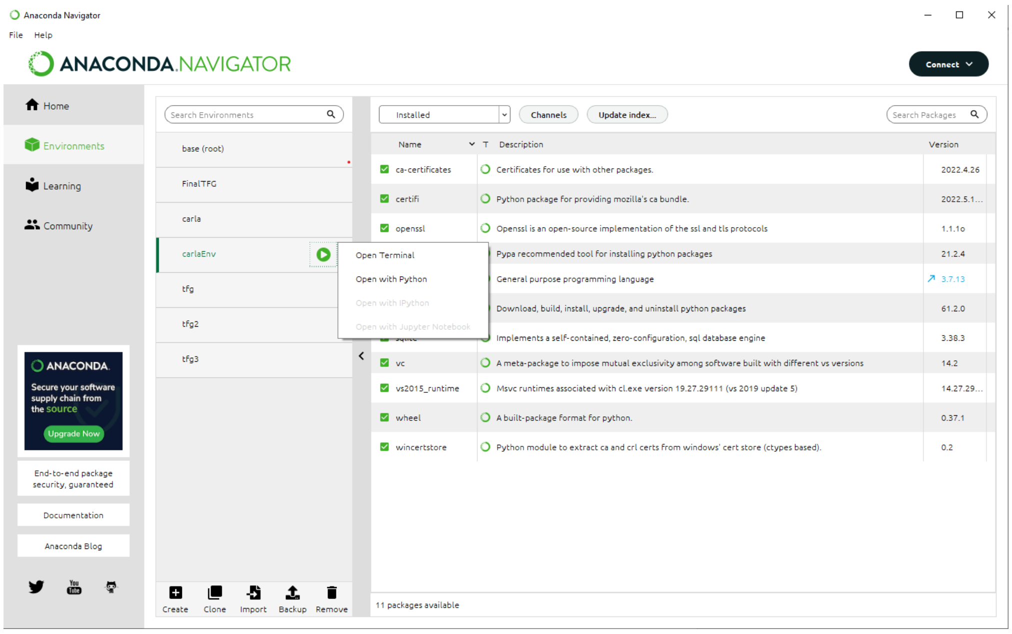
PyCharm Professional
A full-fledged IDE by JetBrains for both Scientific and Web Python development. Supports HTML, JS, and SQL.

RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Figure 7: Anaconda Navigator Home Tab. Source: Own compilation

10.3 Environment Libraries

Once we have finished with the previous two steps, we can start with the installation of the environment libraries. The easy management of environments is one of Anaconda's advantages. First, we go to the Environments tab and select the play button on our previously created environment as shown in the following Figure 8. Then, we select the "Open Terminal" option. In this case, our newly created environment is "carlaEnv".



Name	T	Description	Version
ca-certificates		Certificates for use with other packages.	2022.4.26
certifi		Python package for providing mozilla's ca bundle.	2022.5.1...
openssl		Openssl is an open-source implementation of the ssl and tls protocols	1.1.1o
		Pypa recommended tool for installing python packages	21.2.4
		General purpose programming language	3.7.13
		Download, build, install, upgrade, and uninstall python packages	61.2.0
[illegible]		Implements a self-contained, zero-configuration, sql database engine	3.38.3
vc		A meta-package to impose mutual exclusivity among software built with different vs versions	14.2
vs2015_runtime		Msvc runtimes associated with cl.exe version 19.27.29111 (vs 2019 update 5)	14.27.29...
wheel		A built-package format for python.	0.37.1
wincertstore		Python module to extract ca and crl certs from windows' cert store (ctypes based).	0.2

Figure 8: Anaconda Navigator Environments Tab. Source: Own compilation

This will open a command prompt (cmd) which has our environment loaded. To install the libraries needed for the correct execution of CARLA we will simply type in the following commands in the correct order:

Step 1, installing Tensorflow 1.14. Depending on this version, all the next steps will vary. If you choose a more updated version other libraries such as Keras will change.

```
conda install tensorflow-gpu==1.14
```

Step 2, installing OpenCV.

```
conda install -c conda-forge opencv
```

Step 3, installing Keras. A compatible version for Tensorflow 1.14 is Keras 2.2.4.

```
conda install keras==2.2.4
```

Step 4, installing tqdm

```
conda install tqdm
```

With these four steps we will be ready to execute CARLA in our machine. To check for bugs, see the Appendix B.2 for the complete list of the installed libraries.

10.4 CARLA

Upon completion of the above. But first we need to understand the key concept of the software. It is composed mainly by two things: a server which is the main program (CARLA.exe) and a client. The power of CARLA simulator resides in its capacity to be controlled with scripts by this external client. It can manipulate most of the aspects of simulation, from environment to duration of each episode. It can retrieve data from different sensors and send control instructions to the player vehicle. Now we can start with the installation of CARLA.

10.4.1 Getting Started

To download CARLA's software go to their website. This will take us to a GitHub [18] where we will find all the existing versions. Here we will select version 0.9.5 as it is the one that has been used for all the implementation and experimentation. We save the file and once downloaded we can start the software execution.

10.4.2 CARLA Server

To run the CARLA server, we have two options. The first is to go to the location where we saved the file and open the directory. Inside we will find several files. Among them will be one called "CARLA.exe" and to run it, simply double click on it. The disadvantage of using this method is that it will open CARLA, specifically the server, with default options.

If we want to modify these rendering options, we will have to recur to the second method of execution. To do this, we will open Anaconda Prompt (the Anaconda CMD) and go to the location of the file we downloaded earlier. After we have accessed the directory, to execute CARLA we could use the following command:

```
./CarlaUE4.exe -quality-level=Epic
```

In this case we have launched the server with a different quality option. The main options that we will be using are:

- Quality option, it has two different levels for graphics quality. Epic quality is the default and it’s the most detailed. Low quality disables all post-processing and shadows. Depending on the GPU card used we will use either low or epic, but to improve performance for the neural network with the Tensorflow and Keras libraries we will be using low quality.

```
./CarlaUE4.exe -quality-level=Low
```

- Resolution option, the application window can be reshaped to any size, but a low resolution is recommended due to the same reason as before.

```
./CarlaUE4.exe -windowed -resx=800 -resy=600 -quality-level=Low
```

- World option, the world map can be changed by console or by script. Typing the map name after the main command will work.

```
./CarlaUE4.exe Town02 -windowed -resx=800 -resy=600 -quality-
level=Low
```

10.4.3 CARLA Client

The client provides a Python module for communicating with the CARLA server. In the folder "PythonAPI" from the downloaded file, three directories will show up with examples of scripts to be executed. In all of them, the client connects to the server through the default port "2000". If the connection fails, code written in any of the scripts will not execute and raise an exception.

To execute a client successfully you will need to run first the server and then execute the python script. For example, if we wanted to execute the "tutorial.py" from the "examples" directory we will need to navigate through with the Anaconda Prompt until we reach the following path:

```
$\CARLA_TFG
```

Run the server:

```
./CarlaUE4.exe Town02 -windowed -resx=800 -resy=600 -quality-
level=Low
```

Then, reach the next path:

```
$\CARLA_TFG\PythonAPI\examples
```

And execute the following python command:

```
python tutorial.py
```

With this last step done, we have achieved our first objective: to create and set up an environment for CARLA. Therefore, we can start the implementation and experimentation of the models.

11. Implementation

In this section we will discuss all the elements that we need to implement to start with the experimentation.

10.1 The World

The first element we need to be certain about is the world. The learning of our car can be affected by the complexity of the world itself. Having a world that is too complicated for the start of the experiments could be counterproductive due to a high number of observations it would receive. That is why in the learning process we will start with a simple world, and eventually we will scale its difficulty by changing the maps. The following figure shows a summary about each map (see Appendix C.1 for more information).

Town	**Summary**
Town 01	As Town 02, a basic town layout with all "T junctions". These are the most stable.
Town 02	As Town 01, a basic town layout with all "T junctions". These are the most stable.
Town 03	The most complex town with a roundabout, unevenness, a tunnel. Essentially a medley.
Town 04	An infinite loop in a highway.
Town 05	Squared-grid town with cross junctions and a bridge.
Town 06	Long highways with a lane exit.
Town 07	A rural environment with narrow roads, barely non traffic lights and barns.

Table 8: Town descriptions. Source: Own compilation

The candidate map that we will be using in the beginning of the experiments is shown in the following Figure 9. This is because it is a small square city with several crossroads and no roundabouts. The roundabouts are usually quite tricky for autonomous vehicles.

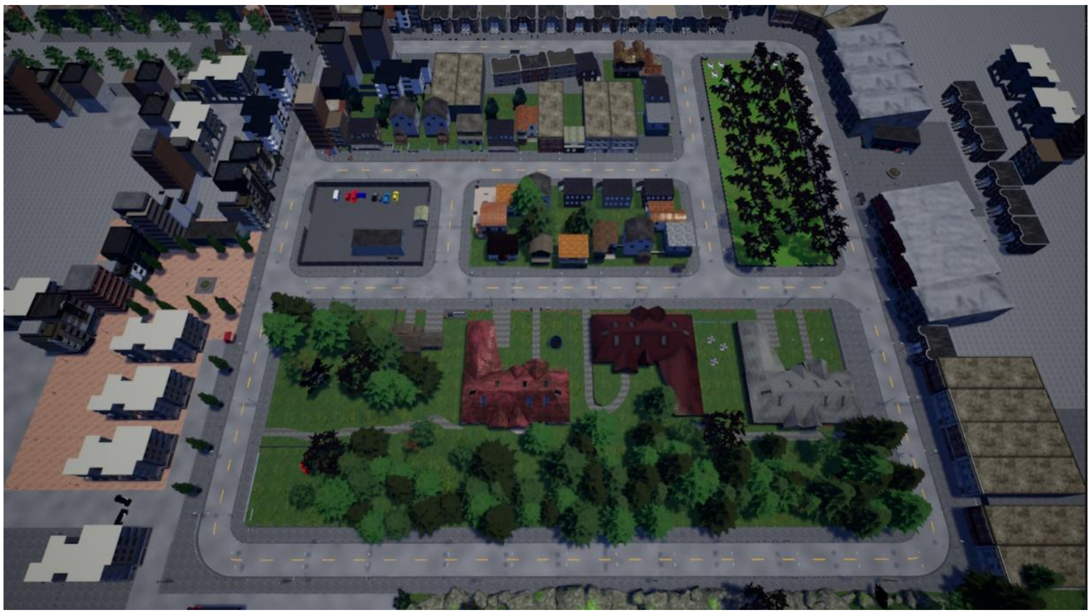


Figure 9: Top view of Town02. Source: Own compilation

10.2 The Car

Actors in CARLA are the elements that perform actions within the simulation, and they can affect other actors. They include vehicles, walkers, sensors, traffic signs, traffic lights and the spectator. It is crucial to have full understanding on how to operate on them.

Figure 10: Tesla Model 3 model. Source: Own compilation

Our testing vehicle will be a Tesla Model 3 and it has several sensors that we can use as inputs for our training. These sensors can be divided into four main groups:

Cameras are the first main type of sensors that we can use in or vehicle. They take a shot of the world from their point of view a can retrieve data in every simulation step.

1. Depth camera renders the depth of the elements in the field of view in a grey-scale map.
2. RGB camera provides clear vision of the surroundings. Looks like a normal photo of the scene.
3. Semantic segmentation camera renders elements in the field of view with a specific colour according to their tags.
4. Instance segmentation camera renders elements in the field of view with a specific colour according to their tags and a unique object ID.

The next group of sensors are **detectors,** and they retrieve data when the object they are attached to registers a specific event.

1. Collision detector retrieves collisions between its parent and other actors.
2. Lane invasion detector registers when its parent crosses a lane marking.
3. Obstacle detector detects possible obstacles ahead of its parent.

The third group embraces all the **other** sensors. They have different functionalities such as navigation, measurement of physical properties and 2D/3D point maps of the scene.

1. GNSS retrieves the geolocation of the sensor.
2. IMU comprises an accelerometer, a gyroscope, and a compass.
3. LIDAR generates a 4D point cloud with coordinates and intensity per point to model the surroundings.
4. RADAR models a 2D map and points the elements in sight regarding the sensor.

The last group contains all the sensors or systems created by me and used by the car.

1. A waypoint system, it was developed to help the car reach the destination point by dividing the route into several points separated by one meter from each other.
2. An off-road detection system, it was created to detect when the car goes out of the road.

During training, we will be changing the sensors to obtain the best model.

10.3 Reinforcement Learning

Reinforcement Learning is all about learning the optimal behaviour in an environment to obtain maximum reward with an agent.

10.3.1 Introduction

In RL we have two main actors: the agent and the environment, in this case CARLA, in which an agent is trained. To do this training possible we have three main interactions between them. These are the following:

- State: it describes the status of the agent. This could be the current position of the car, the speed or even the orientation of the vehicle.

- Action: it describes what can an agent do. Moving forward, pressing the brake, or rotating the car are different actions that can be performed.

- Reward: it is the feedback on the action previously taken on a state. This could be a positive or negative reward.

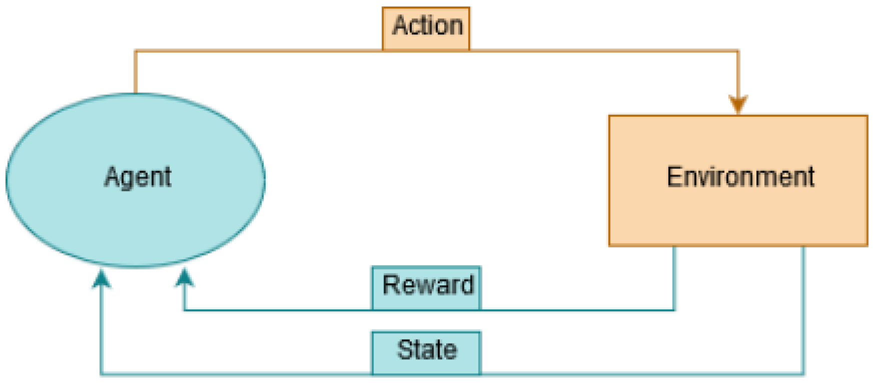


Figure 11: Agent-Environment Interaction Diagram. Source: Study of the implementation of an autonomous driving system, Marta Basquens [19]

Maximizing the reward obtained by performing a series of actions is the main objective of the agent as we previously mentioned. Therefore, a proper reward system is very important as choosing one action or the other will be affected by it.

During the training process we will see two main stages: exploration and exploitation [20]. The exploration stage has a specific purpose, to visit states that you have not yet visited or to take actions you have not yet taken. When you start the first episodes you want the agent to explore and do some trial and error to find the actions that will return a positive reward. On the other hand, the exploitation stage is when you decide to use the knowledge and memory from the previous episode to maximise the rewards. This stage usually comes at the end of the training session or when you have a trained agent.

The agent will require a policy to learn. The policy defines how the agent will behave at a given time and map from perceived states of the environment to actions to be taken when in those states. To store all this data the policy uses a huge table with all the information saved for each state. At the beginning, if the agent has not been trained before this table will be empty, in other words, it will have no experience. This in when the exploration stage becomes useful. After the agent has performed a series of steps, it will start leaning by adjusting its predicted rewards for specific pairs of state-actions towards the received rewards. These predicted values are known as Q-values and are used as feedback on how good a state is.

The parameter that determines the probability of the agent performing a random action is the epsilon ($\in$). It is a real value that exists in [0, 1]. This variable introduces randomness into our training algorithm forcing the agent to try different actions. A high epsilon, for example, $\in = 1$ will result in all actions are random which means we are in the exploration stage. On the other hand, a lower epsilon such as $\in = 0$ will result in no random actions which means we are in the exploitation stage. Implementing an epsilon decay formula is very important during the training process. This is because we want to travel from the exploration stage to the exploitation to maximize learning. The formula we will be using will be the following:

$$\in_n = \in_{n-1} \times \in_{decay}$$

We will need to set an $\in_{decay}$ that, when multiplied by the current epsilon, will give us the next epsilon value. To calculate the epsilon value in a certain number of episodes we will use the following formula:

$$\in = \in_{initial} \times \in_{decay}^{n}$$

Where n is the number of episodes, $\in_{decay}$ the epsilon decay per episode and $\in_{initial}$ the epsilon used at the beginning.

Another important parameter is the discount factor (γ). It is a real value that exists in [0, 1] and determines how the agent should care about the rewards achieved in the past, present and future. If γ=0 the agent will focus on actions that produce an immediate reward whereas if γ=1 the agent cares on actions that will produce a high future reward due to a sequence of actions.

10.3.2 DQN Agent

Deep Q-Network is a reinforcement learning algorithm that combines Q-Learning with deep neural networks to let RL work for complex, high-dimensional environments, like video games, or robotics.

DQN uses the Bellman’s equation to solve the problem of being unable to see future rewards. Otherwise, only immediate rewards would be shown. The formulation of the equations is as follows:

$$Q^{new}(s_t, a_t) = (1-\in) \times \underbrace{Q(s_t, a_t)}_{old\ value} + \in \times \underbrace{(r_t + \gamma * max(Q^{new}(s_{t+1}, a)))}_{learned\ value}$$

Equation 1: Bellman's equation

Where:

- Q is the accumulated reward value
- t is the current state
- s is a specific state
- a is a specific action
- $\in$ is the epsilon, also known as learning rate.
- r is the reward associated to a specific state
- γ is the discount factor

The agent can use this equation to link up all the information learned resulting in a batch of memory that can be used for improving the efficiency of its learning.

10.3.3 ANN

The artificial neural network used in this project is recycled from Marta's Thesis, Study of the implementation of an autonomous driving system. We used Marta's model because she previously carried out a study in her thesis on which artificial neural network gave the best result. In her case, she decided to use a CNN (convolutional neural network):

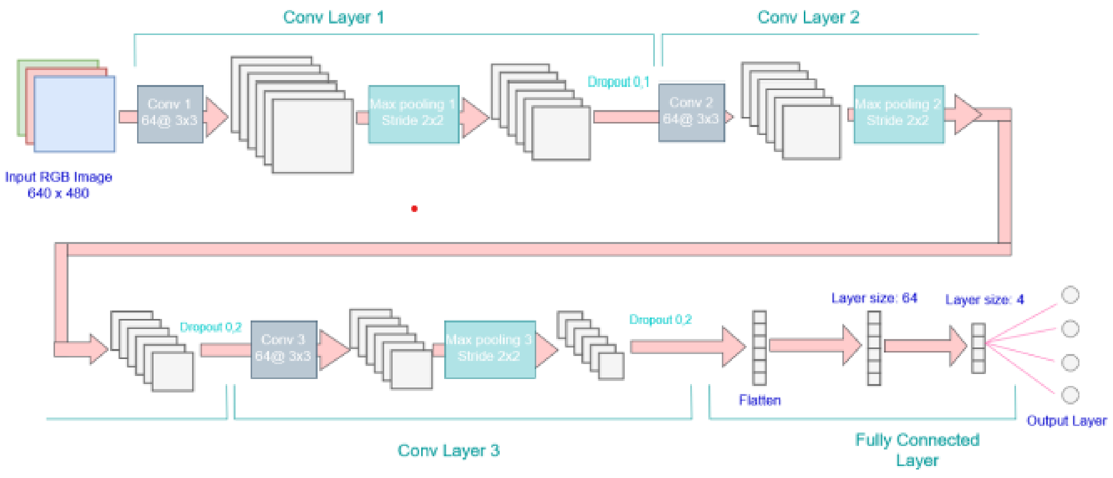


Figure 12: Structure of the Neural Network used in this project. Source: Study of the implementation of an autonomous driving system, Marta Basquens

It consists of:

- 3 convolutional layers with 64 neurons each + ReLU activation function which is used for filtering information.
- 3 pooling layers performed with max average. They are used to reduce the dimensions of the feature maps in this case 2x2.
- 3 dropouts layers following the convolutional networks to avoid overfitting.
- 1 flatten layer to convert the vector to 1D.
- 1 dense layer with 64 neurons.
- 1 dense layer with 4 neurons (4 actions) with linear activation function. It is used to determine the output of neural network like yes or no.

10.3.4 Actions and Rewards

The initial and final actions for the training cases are the following:

ID	Action
1	Accelerate car with throttle
2	Decelerate car with brake
3	Turn right at 90º
4	Turn left at 90º

Table 9: Actions used by the agent. Source: Own compilation

We decide to keep it up during all the thesis due to simplicity. Changing the number of actions would cause a modification in the structure of the artificial neural network. This change prevents us from continuing with the training of an experienced agent if it had previously been trained with a different number of actions. Furthermore, the training of the agents is very long, and we cannot afford to train several agents from scratch, as obtaining a result would take several days of training.

The initial reward system is also recycled from Marta's final reward system. We used her system as a basis since it would be our starting point to begin with.

ID	Reward	Reward
1	Collision with an object	-20 points
2	Velocity bigger or greater than 50 km/h	+3 points
3	Velocity smaller than 50 km/h	-2 points
4	More than 10 seconds without a collision	+5 points
5	No movement	-8 points
6	Short training episode	-10 points

Table 10: Initial reward system used by the agent. Source: Own compilation

During the training experiments we changed the reward system in several ways but concluded that using a formula would give us a better outcome. The formula has been inspired by the Learning to Drive in a Day by Kendall [21] and the Learning to Drive Smoothly in Minutes by Raffin [22] and is as follows:

$$r\,(v,d) = \begin{cases} 10 & \text{on reached destination} \\ -10 & \text{on infraction} \\ \dfrac{v}{v_{min}} \times (1 - d_{norm}) \times \alpha_{rew} & v < v_{min} \\ 1 \times (1 - d_{norm}) \times \alpha_{rew} & v_{min} \le v < v_{target} \\ \left(1 - \dfrac{v - v_{target}}{v_{max} - v_{target}}\right) \times (1 - d_{norm}) \times \alpha_{rew} & v \ge v_{target} \end{cases}$$

Equation 2: Multiplied centering, angle, and speed rewards formula. Source: Own compilation

Where:

- v is the car actual speed.
- v_{target} is the car target speed.
- v_{max} is the car maximum speed.
- v_{min} is the car minimum speed.
- d is the distance from the centre of the road.
- d_{norm} is the distance from the centre of the road interpolate from1 when centred to 0 when 3 meters from the centre.
- α_{rew} is the reward obtained by the interpolation from 1 when aligned with the road to 0 when +/- 20 degrees of road.

The reward for each step has been normalize and will only return a value interpolated from 0 when performed bad to 1 when performed excellent unless it has committed an infraction or reached destination. This normalization has helped the car obtained better results as now rewards are more stable.

12. Experimentation

12.1 Training Model

A training model is a dataset used to train a machine learning algorithm. The parameters we will watching throughout all the trainings are accuracy, epsilon decay, loss value, average reward, maximum reward, and minimum reward. The accuracy is the ratio of number of correct predictions to the total number of input samples. The loss value implies how well or poorly a certain model behaves after each iteration of optimization. Ideally, one would expect the reduction of loss after each episode. The other four parameters are self-explanatory.

12.1.1 Training 1: Marta's Model

The first training will be done in "Town02", and we will be using Marta's model with the rewards and actions she implemented in her thesis. This test is the first approach with CARLA and all its environment set-up. Obviously, good results are not going to be expected in this iteration, only a simple replica of her final project.

The ideal goal would be to teach the car to stay on the road and try to make the car able to get at least once from point A to point B. The agent won't have a specific route to travel or even a waypoint system. In the following figure we can see the start point and the destination point.

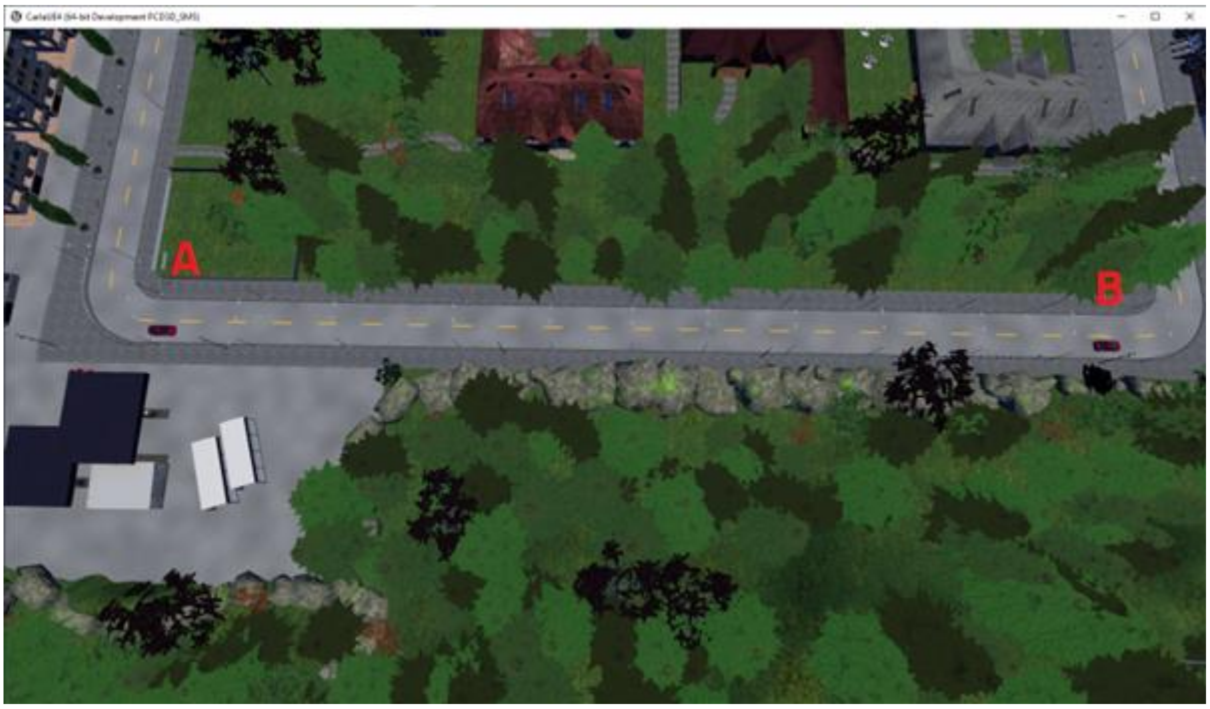


Figure 13: First route in Town02. Source: Own compilation

<ins>Training Conditions:</ins>

Training Settings			
Total Training Time	2h 39m 42s	Episodes	1000
Town	Town02	Seconds Per Episode	9.58
Actions	Action 1: Accelerate car with throttle Action 2: Turn left at 90º Action 3: Turn right at 90º Action 4: Decelerate car with brake		
Rewards	Reward 1: Collision: -20 points Reward 2: Velocity > 50 km/h: +3 point Reward 3: Velocity < 50 km/h: -3 point Reward 4: > 10 seconds without collision: +5 points Reward 5: No movement: -8 points Reward 6: Short training episode (<= 10 seconds): -10		
Sensors Used	Semantic Segmentation camera Collision detection sensor		

Table 11: Settings for training 1. Source: Own compilation

<ins>Results</ins>

Thanks to Tensorboard we can extract a series of graphs that we can use to obtain some benefits. (To see all the graphs generated see Appendix D).

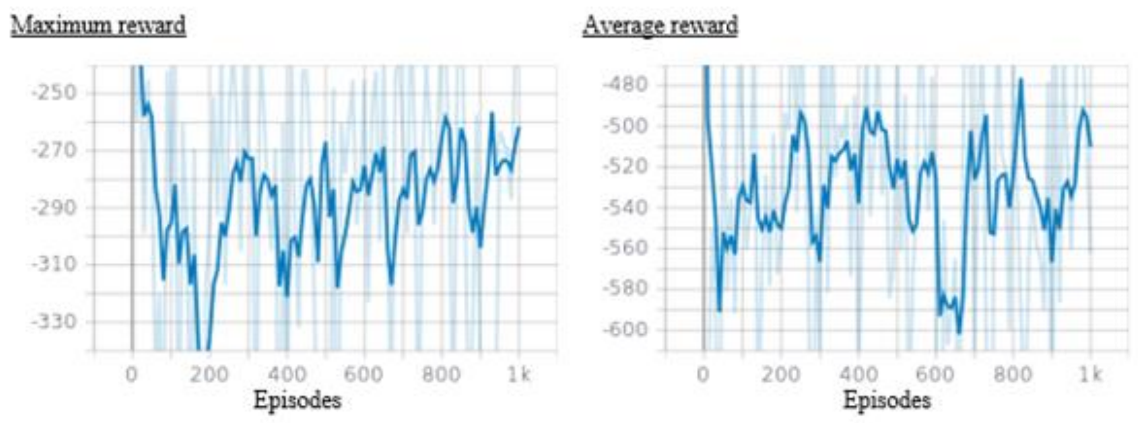


Figure 14: Maximum and Average Reward from Training 1. Source: Own compilation

Discussion

The graphs show that during the training session, the maximum and average reward had an upward trend. This is a good sign as it shows that the car is learning something and trying to maximise this reward. The problem is that when we look at the learning episodes themselves, we realise that the car is driving but is not aware that it must get to a particular place. Moreover, it does not consider the directions of the lane, the car simply decides to stay on the road.

From this experiment we can draw the following conclusion, the agent needs a waypoint system and a new reward system.

12.1.2 Training 2: New Model with Waypoint System

For this training it was decided to implement a new reward system as the previous one did not consider anything about reaching a desired destination. In addition, a new waypoint system was created which consists of dividing a trip from point A to point B into small one-metre sections. The actions remained the same.

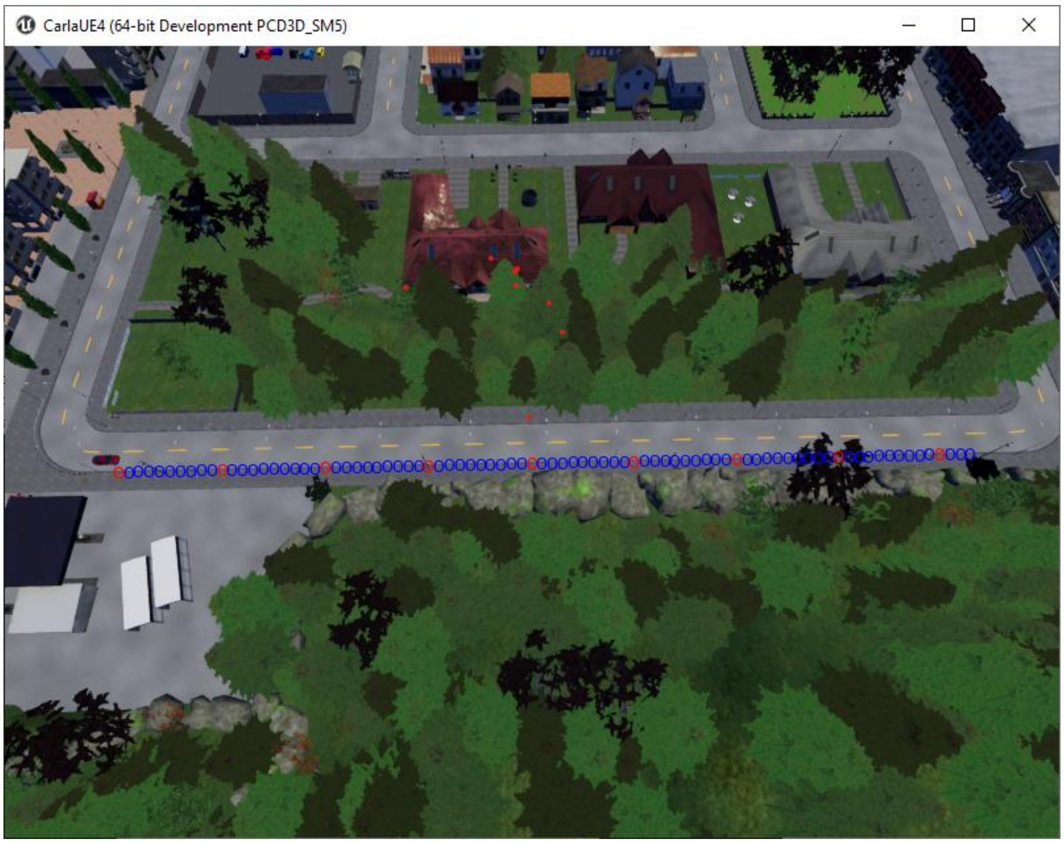


Figure 15: Route in Town02 for Training 2. Source: Own compilation

<u>Training Conditions:</u>

Training Settings			
Total Training Time	4h 17m 15s	Episodes	1500
Town	Town02	Seconds Per Episode	10.29
Actions	Action 1: Accelerate car with throttle Action 2: Turn left at 90º Action 3: Turn right at 90º Action 4: Decelerate car with brake		

Training Settings	
Rewards	Reward 1: Collision: -20 points Reward 2: Arrived at Destination: +20 points Reward 3: If distance to destination increases: -5 Reward 4: If distance to destination increases: +5 Reward 5: Short training episode (<= 20 seconds): -10
Sensors Used	Semantic Segmentation camera Collision detection sensor Waypoint system

Table 12: Settings for training 2. Source: Own compilation

Results

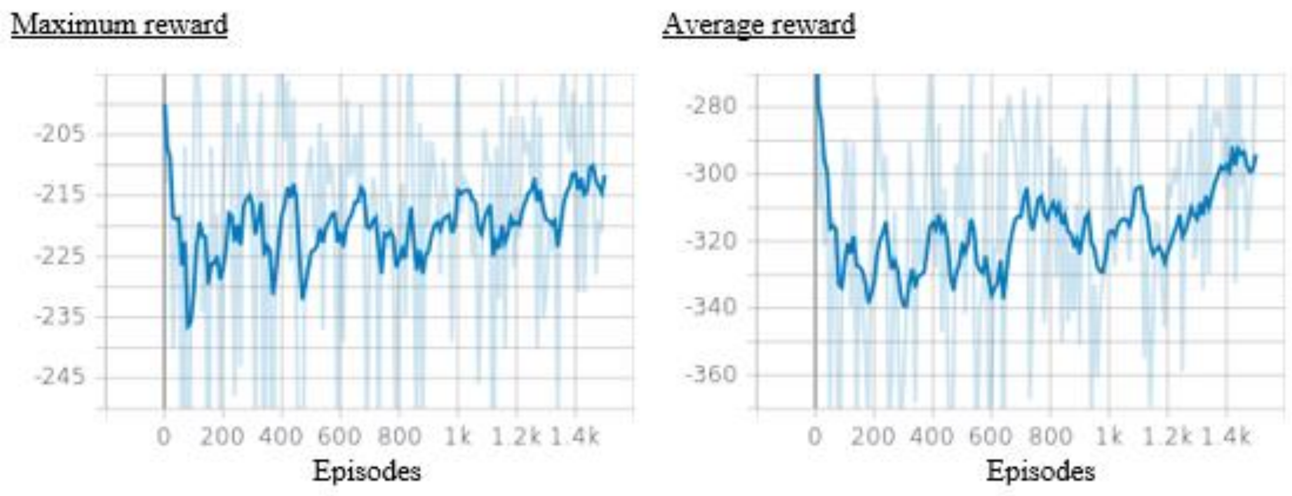


Figure 16: Maximum and Average Reward from Training 2. Source: Own compilation

Discussion

In this training we can see a great improvement in the episodes. Now we can see a big upward trend in the graph of the average reward and in the maximum reward. This indicates that the car itself is learning to get better rewards. At this point, the car has a very simple waypoint system, at each step the system returns the number of meters between the car and the destination. If the distance is reduced, a reward is given and if it is increased, a penalty is given. We will probably have to change this waypoint system as it is correct for straights but for more complex routes can be a problem.

Also, the car has learnt to accelerate earlier as it has realised that if it goes in a straight line faster it gets a higher score. One problem we have found is that the car is not able to realise when it leaves the track. It only focuses on reducing the distance from the destination point and sometimes takes forbidden paths.

12.1.3 Training 3: Waypoint System + Off-Road Detection System

For this training session we have kept the same waypoint system, although we have said that it is still quite simple, and decided to add a new sensor, the off-road detection system. Now the car will know when it goes off-road.

The number of episodes was increased to 4000 as we wanted to give the car more training time. One of the biggest problems we encountered is the number of episodes and the training time. CARLA is a high demanding software, and my computer can't run two agents at the same time or even run it at a higher speed as the computer cannot handle the requirements. The reward system and actions remain the same.

Training Conditions

Training Settings			
Total Training Time	9h 25m 20s	Episodes	4000
Town	Town02	Seconds Per Episode	8.48
Actions	Action 1: Accelerate car with throttle Action 2: Turn left at 90º Action 3: Turn right at 90º Action 4: Decelerate car with brake		
Rewards	Reward 1: Collision: -20 points Reward 2: Arrived at destination: +20 points Reward 3: If distance to destination increases: -5 Reward 4: If distance to destination increases: +5 Reward 5: Short training episode (<= 20 seconds): -10		
Sensors Used	Semantic Segmentation camera Collision detection sensor Waypoint system Off-Road detection system		

Table 13: Settings for training 3. Source: Own compilation

Results

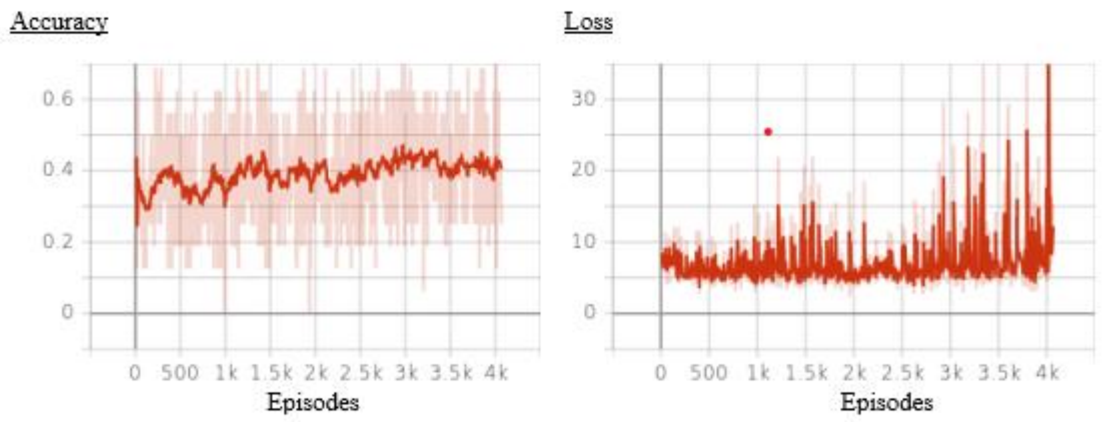


Figure 17: Accuracy and Loss for Training 3. Source: Own compilation

Discussion

This time we decided to focus mainly on the parameters of accuracy and loss. During training, the tracking of the rewards was not saved correctly, and the rewards were only stored for episodes 0, 2000 and 4000. We decided not to repeat the training as it would be almost 10 hours wasted and you can always learn from your mistakes. In this case, this error made us look more closely at other parameters. The accuracy of the model was quite low, normally, a good model should be around 0.7. Also, the loss value was behaving incorrectly, as it should reduce as the episodes go by. Therefore, we concluded that we were not getting it right with the new modifications. We still decided to test how the car would look on a new route to confirm that this was indeed the problem.

12.1.4 Training 4: New Route

This training had to be repeated because during the first one the laptop shut down due to temperature problems. I had run about 1000 episodes. From then on, we started to be more careful with the training sessions and the ventilation of the laptop itself was improved.

The only difference in this training is that we changed the route of the car as you can see in the following figure. The car is facing its first turn.

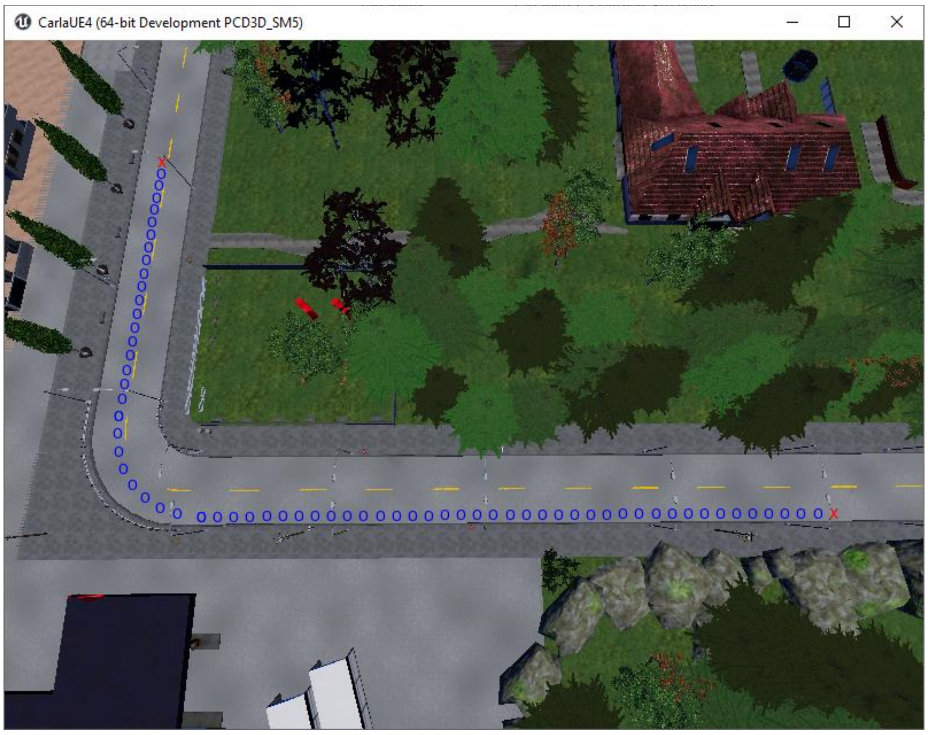


Figure 18: Route in Town07 for Training 4. Source: Own compilation

<u>Training Conditions</u>

Training Settings			
Total Training Time	4h 57m 39s	Episodes	2000
Town	Town02	Seconds Per Episode	8.93
Actions	Action 1: Accelerate car with throttle Action 2: Turn left at 90º Action 3: Turn right at 90º Action 4: Decelerate car with brake		

Training Settings	
Rewards	Reward 1: Collision: -20 points Reward 2: Arrived at destination: +20 points Reward 3: If distance to destination increases: -5 Reward 4: If distance to destination increases: +5 Reward 5: Short training episode (<= 20 seconds): -10
Sensors Used	Semantic Segmentation camera Collision detection sensor Waypoint system Off-Road detection system

Table 14: Settings for training 4. Source: Own compilation

Results

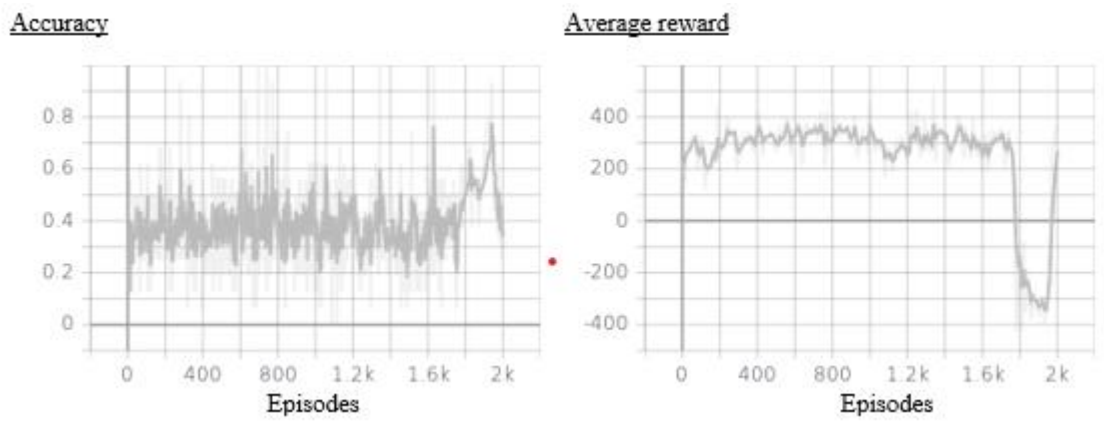


Figure 19: Accuracy and Average Reward from Training 4. Source: Own compilation

Discussion

In this case we will focus on the accuracy and average reward charts. The accuracy graph shows some peaks of 0.7 of accuracy at the end of the training and a drop in the average reward which may suggest the change in the accuracy.

Nevertheless, these rewards were not giving us good results so we will change the reward system again in the next training and implement a new advanced waypoint system to force the car to get the curve right.

12.1.5 Training 5: New Reward System + New Waypoint System

In this training we added a new reward and waypoint system. For the first system we decided to implement a new formula, the multiplied centering angle and speed rewards formula (see Section 10.3.4). It will only return a value between 0 and 1 unless an infraction was committed, or the destination was reached. This normalisation of the rewards allows us to obtain more stable results.

For the new waypoint system, we decided to divide the desired route into small waypoints of one meter each. This is because in old training sessions the car only focused on getting to the destination as fast as possible. This meant that the car would always try to go in a straight line and not follow the road's path.

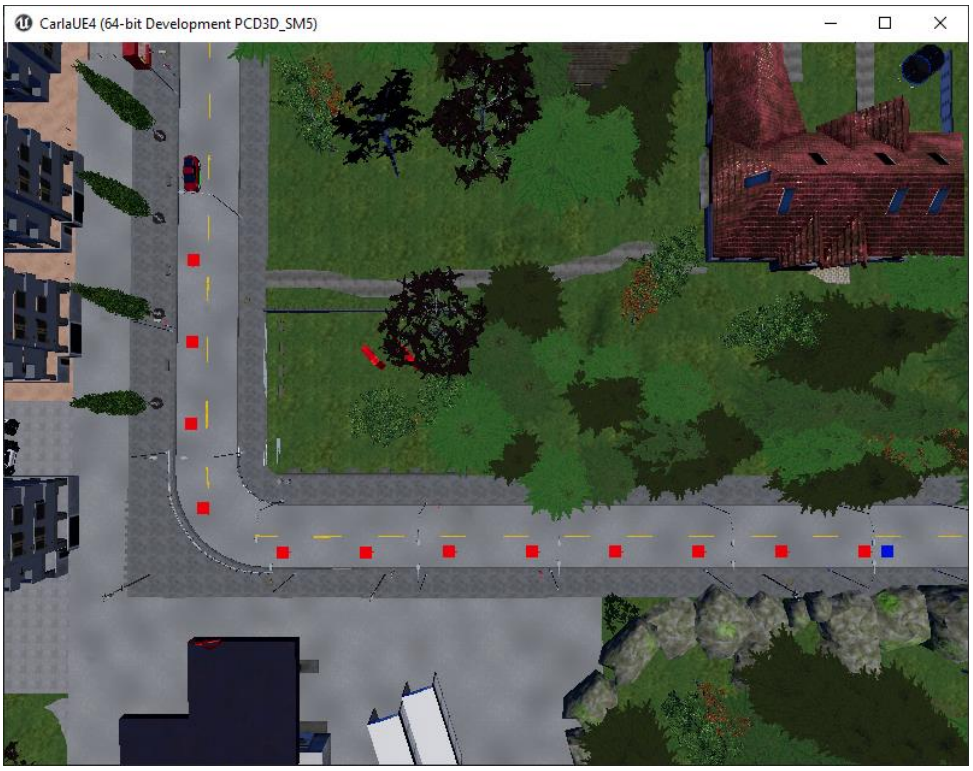


Figure 20: Route in Town02 for Training 5. Source: Own compilation

<u>Training Conditions</u>

Training Settings			
Total Training Time	2h 21m 22s	Episodes	1000
Town	Town02	Seconds Per Episode	8.48s

Training Settings	
Actions	Action 1: Accelerate car with throttle Action 2: Turn left at 90º Action 3: Turn right at 90º Action 4: Decelerate car with brake
Rewards	Reward 1: Infraction: -10 points Reward 2: Reached destination: 10 points Reward 3: Multiplied centering angle and speed rewards formula
Sensors Used	Semantic Segmentation camera Collision detection sensor Waypoint system Off-Road detection system

Table 15: Settings for training 5. Source: Own compilation

Results

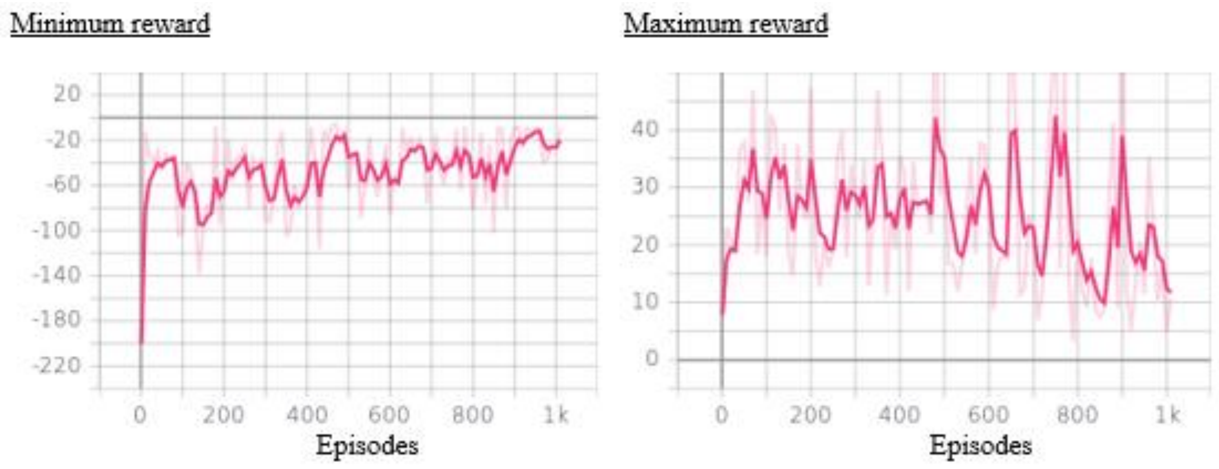


Figure 21: Minimum and Maximum Reward from Training 5. Source: Own compilation

Discussion

With this new reward and waypoint system we see how the car starts to improve in the episodes and especially in the last few episodes it seems to start driving better. If we focus on the minimum reward graph, we can see that there is an upward trend. But in the maximum reward graph we have a downward trend.

One of the infringement conditions was to stay more than 5 seconds with the car stopped, this was a problem because to reset the environment in each episode, sometimes it took about 4 seconds and therefore if the car remained 1 second more stopped the agent detected it as an error. This had to be changed as it often didn't give the agent time to start the car.

12.1.6 Training 6: Update in Rewards

In this training we decided to change one of the infringement conditions as previously mentioned. A total of 4000 episodes were performed and all condition settings remained the same.

Training Conditions

Training Settings			
Total Training Time	7h 09m 31s	Episodes	4000
Town	Town02	Seconds Per Episode	6.44
Actions	Action 1: Accelerate car with throttle Action 2: Turn left at 90º Action 3: Turn right at 90º Action 4: Decelerate car with brake		
Rewards	Reward 1: Infraction (modified): -10 points Reward 2: Reached destination: 10 points Reward 3: Multiplied centering angle and speed rewards formula		
Sensors Used	Semantic Segmentation camera Collision detection sensor Waypoint system Off-Road detection system		

Table 16: Settings for training 6. Source: Own compilation

Results

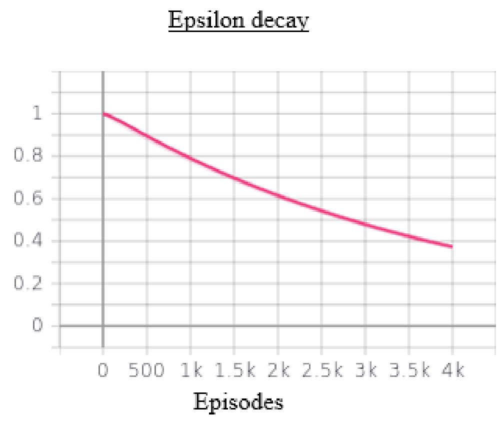


Figure 22: Epsilon Decay from Training 6. Source: Own compilation

<u>Discussion</u>

In this training I spent a lot of time carefully analysing the actions that the car was taking, since throughout the training, many actions seemed random to me. That is why after several episodes of analysis I concluded that the car still took many random actions due to the epsilon parameter.

This parameter oversees determining with what probability an action will be random. During all the previous tests, the epsilon did not go below 0.4 in the last episodes. In other words, the car took 40% of random actions. Obviously, this was a problem as this parameter had not been considered due to inexperience and could be the cause of these inaccurate results.

Despite all this, the car presented a curious behaviour. When arriving at the turn, the car braked drastically as it had learned to slow down in the turns.

12.1.7 Training 7: Epsilon Decay New Policy

For this training we have decided to focus on the epsilon parameter and implement a decay formula as it will allow us to switch from the exploration stage to the exploitation stage. A total of 2000 episodes were performed while maintaining the navigation and reward system.

<u>Training Conditions</u>

Training Settings			
Total Training Time	3h 43m 16s	Episodes	2000
Town	Town02	Seconds Per Episode	6.70
Actions	Action 1: Accelerate car with throttle Action 2: Turn left at 90º Action 3: Turn right at 90º Action 4: Decelerate car with brake		
Rewards	Reward 1: Infraction (modified): -10 points Reward 2: Reached destination: 10 points Reward 3: Multiplied centering angle and speed rewards formula		
Sensors Used	Semantic Segmentation camera Collision detection sensor Waypoint system Off-Road detection system		

Table 17: Settings for training 7. Source: Own compilation

<u>Results</u>

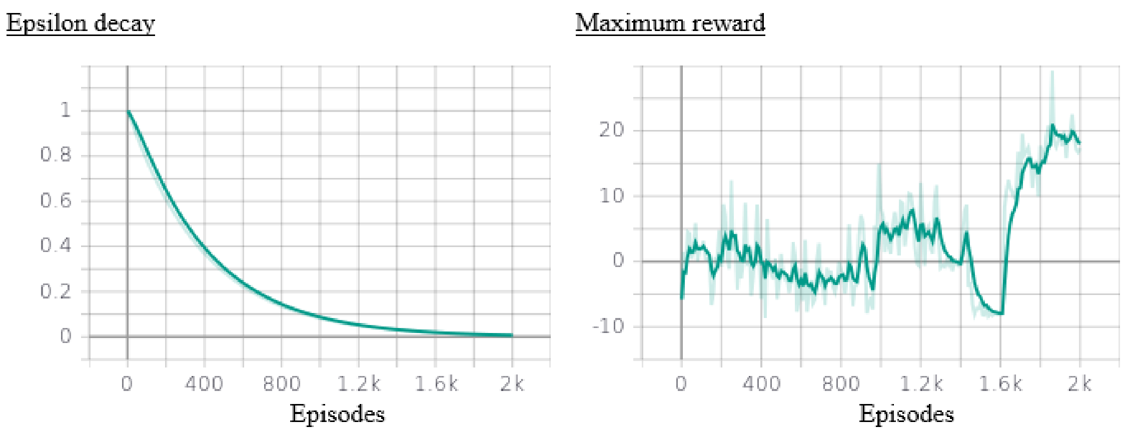


Figure 23: Epsilon decay and Maximum Reward from Training 7. Source: Own compilation

<u>Discussion</u>

In the results we can see how the epsilon drops to values lower than 0.05. This means that we have correctly implemented the epsilon decay formula and in the last episodes the car uses what it has learned to try to obtain the maximum score.

In fact, in this training we can start to see some promising results as the car stays in the lane and is starting to make a turn correctly. It is possible that due to the sharp angle of the turn the car is not learning how to go through it properly. It is worth highlighting that on iterations close to 1500 there is a drastic drop in rewards. This includes minimum reward, maximum reward, and average reward. This may have been due to the laptop having a performance failure.

12.1.8 Training 8: New Map

In this training we decided to change the map since one of the conclusions we reached in the previous training was that sharp turns can confuse the car. The new route to complete the car was simpler (Figure 24). We also decided to increase the number of episodes as we were getting closer to the optimal configurations, and we wanted to see a longer training. Therefore, we set a total of 10000 episodes.

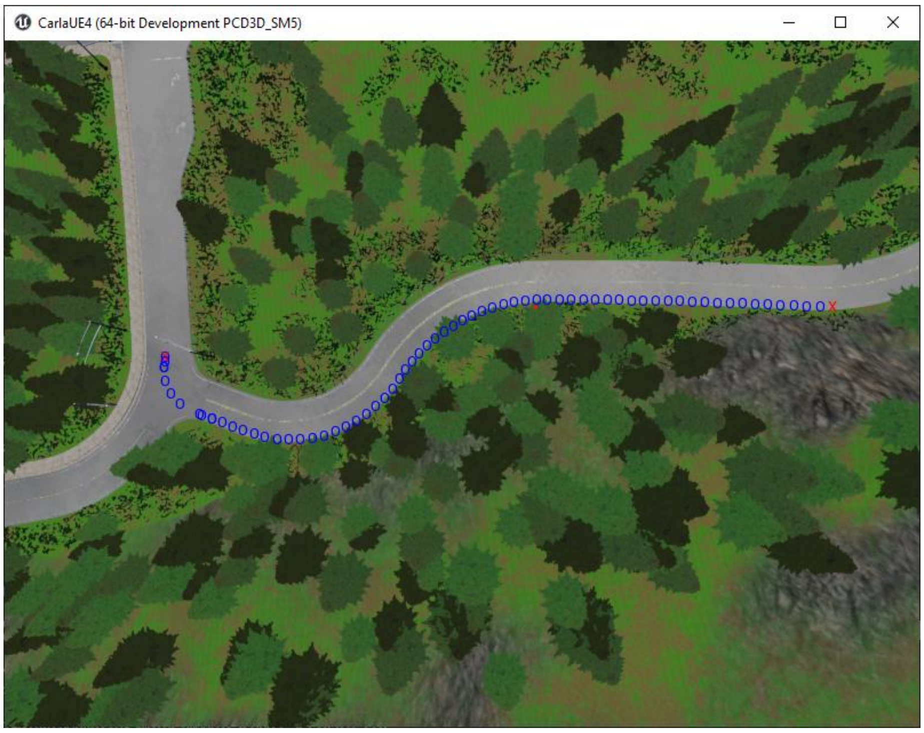


Figure 24: New Route from Town07. Source: Own compilation

<u>Training Conditions</u>

Training Settings			
Total Training Time	23h 25m 00s	Episodes	10000
Town	Town07	Seconds Per Episode	8.34
Actions	Action 1: Accelerate car with throttle Action 2: Turn left at 90º Action 3: Turn right at 90º Action 4: Decelerate car with brake		
Rewards	Reward 1: Infraction (modified): -10 points Reward 2: Reached destination: 10 points		

Training Settings	
	Reward 3: Multiplied centering angle and speed rewards formula
Sensors Used	Semantic Segmentation camera Collision detection sensor Waypoint system Off-Road detection system

Table 18: Settings for training 8. Source: Own compilation

Results

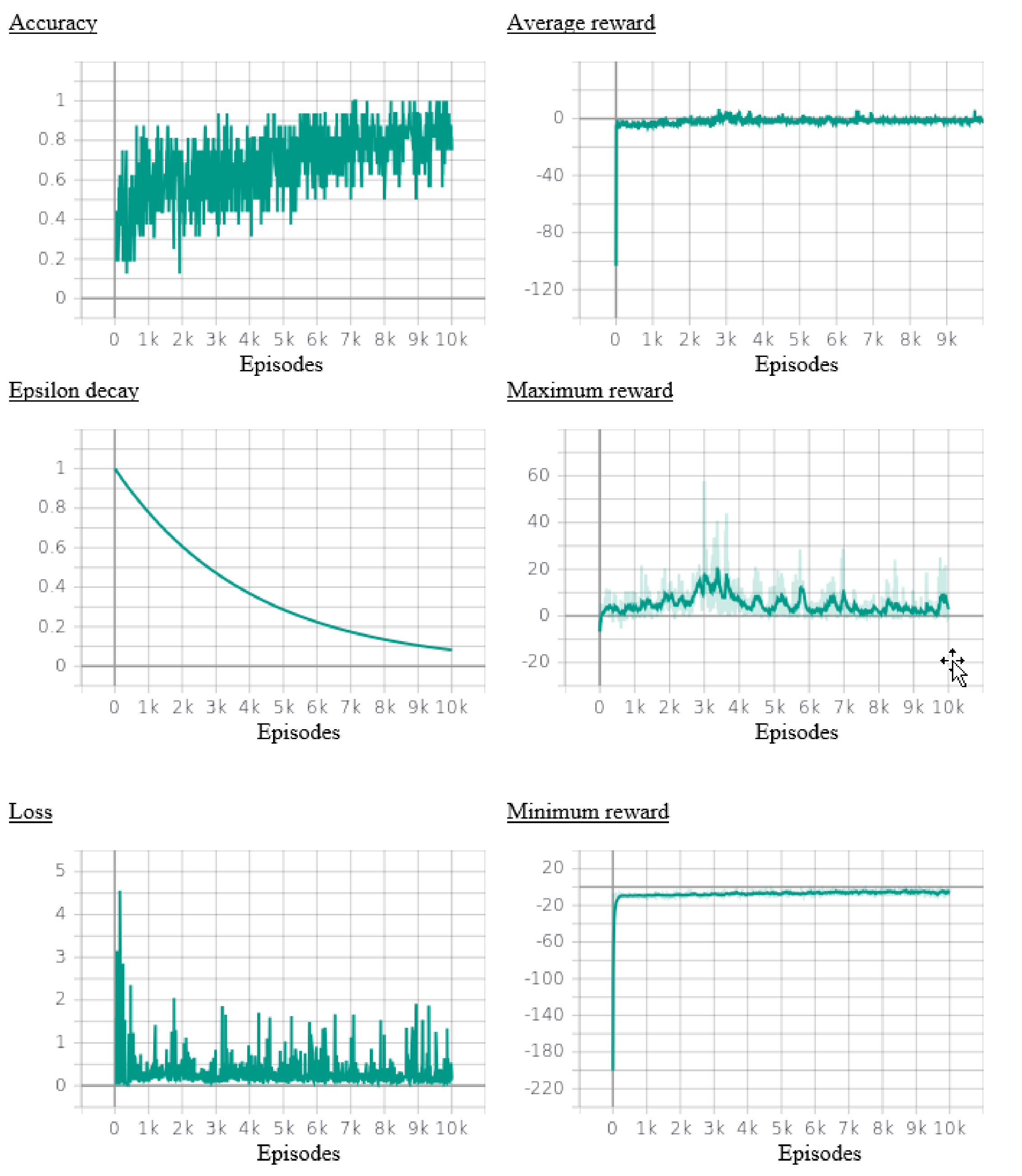


Figure 25: Graphs obtained from the Training 8. Source: Own compilation

Discussion

In this training we will comment all the graphs since it is the last one. First, the accuracy is increasing along all the episodes. This is a good sign because we are approaching a decent model. The epsilon decreases properly, it reaches values lower than 0.1 and therefore the car uses what it has learned. The loss explodes at the beginning but then stabilizes which gives us to understand that the model used is working well. The average reward increases over the episodes, this is because the car starts to learn the route and cannot improve any more. The maximum reward remains stable throughout the iterations. Finally, the minimum reward is increasing as the episodes go by. Overall, the results of this model are good, but if we focus on the practical part, the car has been able to learn the route. This is a great victory since we have completed the main objective of this thesis: that the car can complete a route given a start and end point. With this training completed we decided to move on to the testing model as the settings will no longer be changed.

12.2 Testing Model

Model testing is referred to as the process where the performance of a fully trained model is evaluated on a testing set. In this case, we will be using the model from the last training. Our prediction is that due to the few training episodes we may see some correct driving behaviour.

12.2.1 Testing 1: Town02 Sharp Turn

For the first test we decided to go back to the Town02 map and try a very tight corner that we had previously failed to turn.

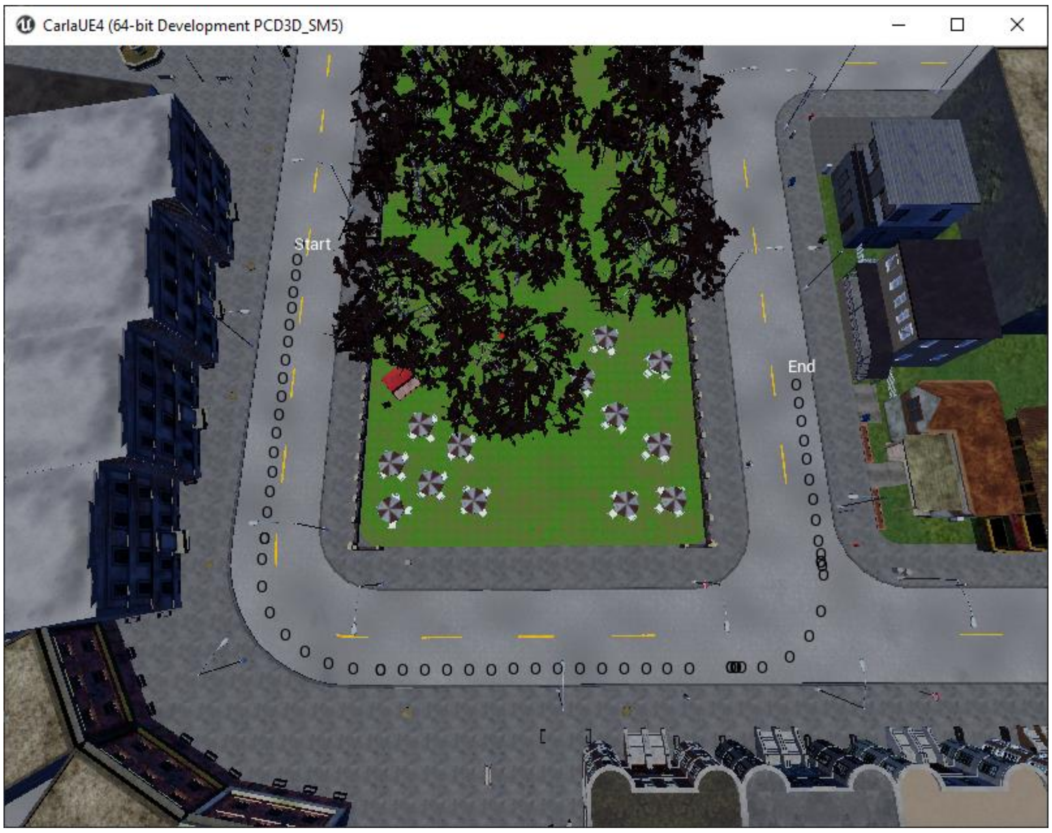
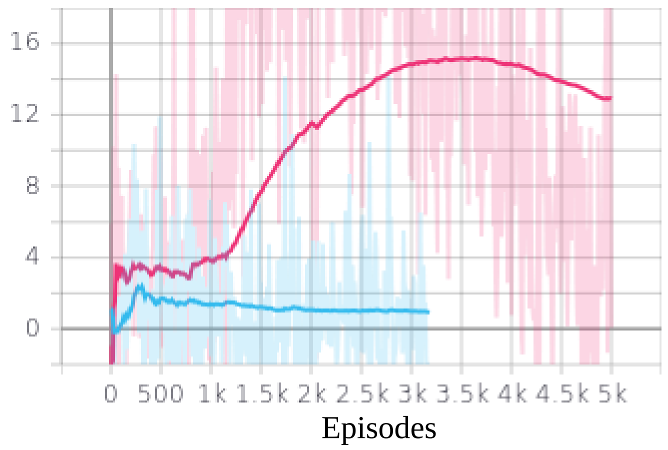


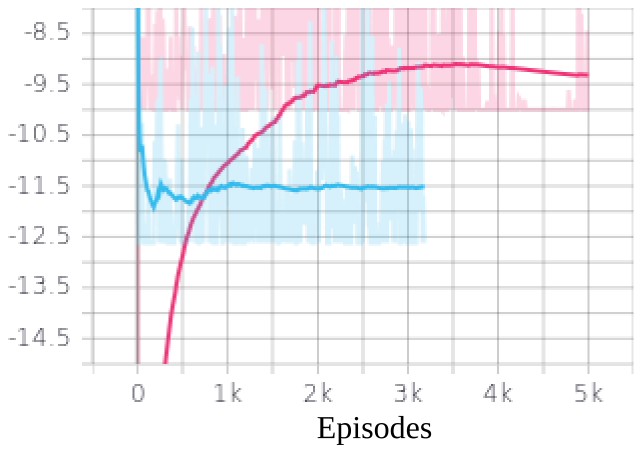
Figure 26: Town02 New Route. Source: Own compilation

We ran several episodes with the Training 8 model for this test. First, we did a test with the discount factor at 0.95 and another one with the value of 0.35. We wanted to see how the agent would behave if we changed this parameter. The results can be seen in Figure 15.

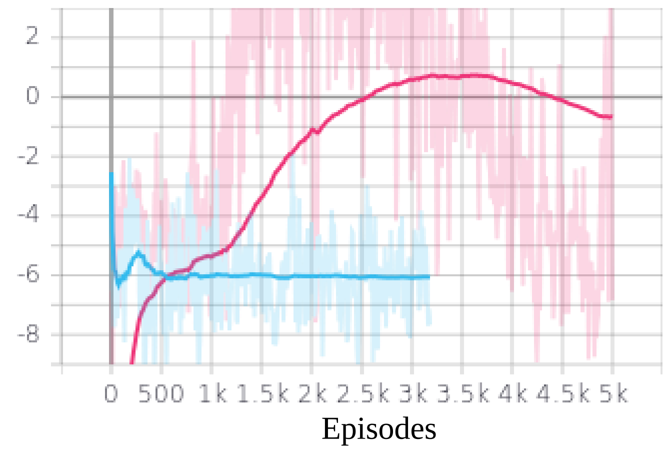
Maximum reward



Minimum reward



Average reward



Legend:

——— : Test A (γ = 0.35)
——— : Test B (γ = 0.95)

Figure 27: Comparison of rewards with different discount factor. Source: Own compilation

During the test, the car stayed on the road, but when it reached the turn, it failed to cross it. This is since the car did not know how to drive through the sharp curve. Not having any previous experience made it impossible for the car to drive through it correctly. However, in this test we concluded that the discount factor at 0.95 gave us better results as it was not looking for immediate reward. It is also worth noting that we had to stop the testing of Test A because it was taking more than 15 hours for only 3000 episodes, meanwhile Test B managed to finish the 5000 episodes in 8 hours.

12.2.2 Testing 2: Town07 Longest Route

This is the last test of the thesis. We switched back to map of Town07 and decided to make a new route. On this route there are different types of curves to see how the car handles itself.

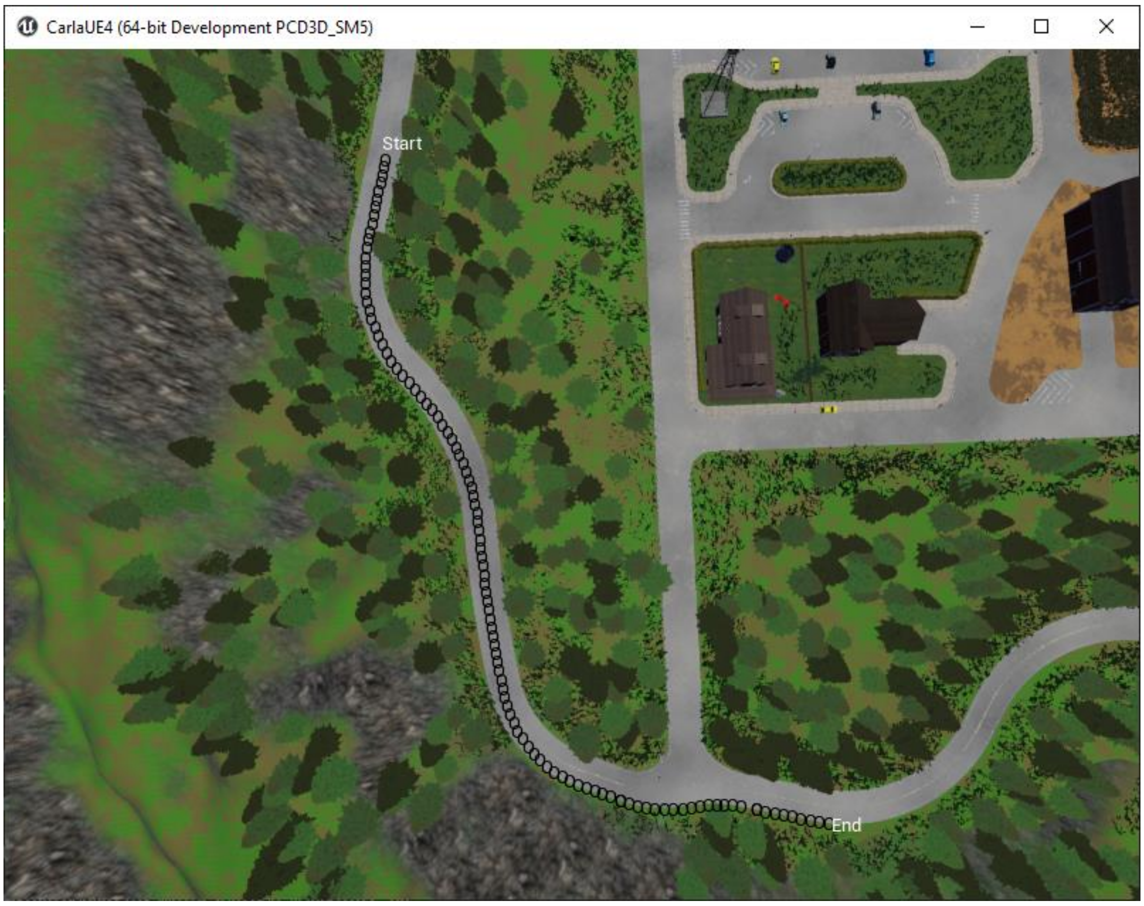


Figure 28: New Route on Town07. Source: Own compilation

In this testing the car managed to complete 80% of the route. My guess is that the car did not know how to interpret the crossroads and therefore was not able to finish it.

If we look at the rewards in Figure 18, we can see from the very beginning the results were quite good. This is due to the previous learning from other occasions. Even though he had experience he was not able to get 100% of the route completed.

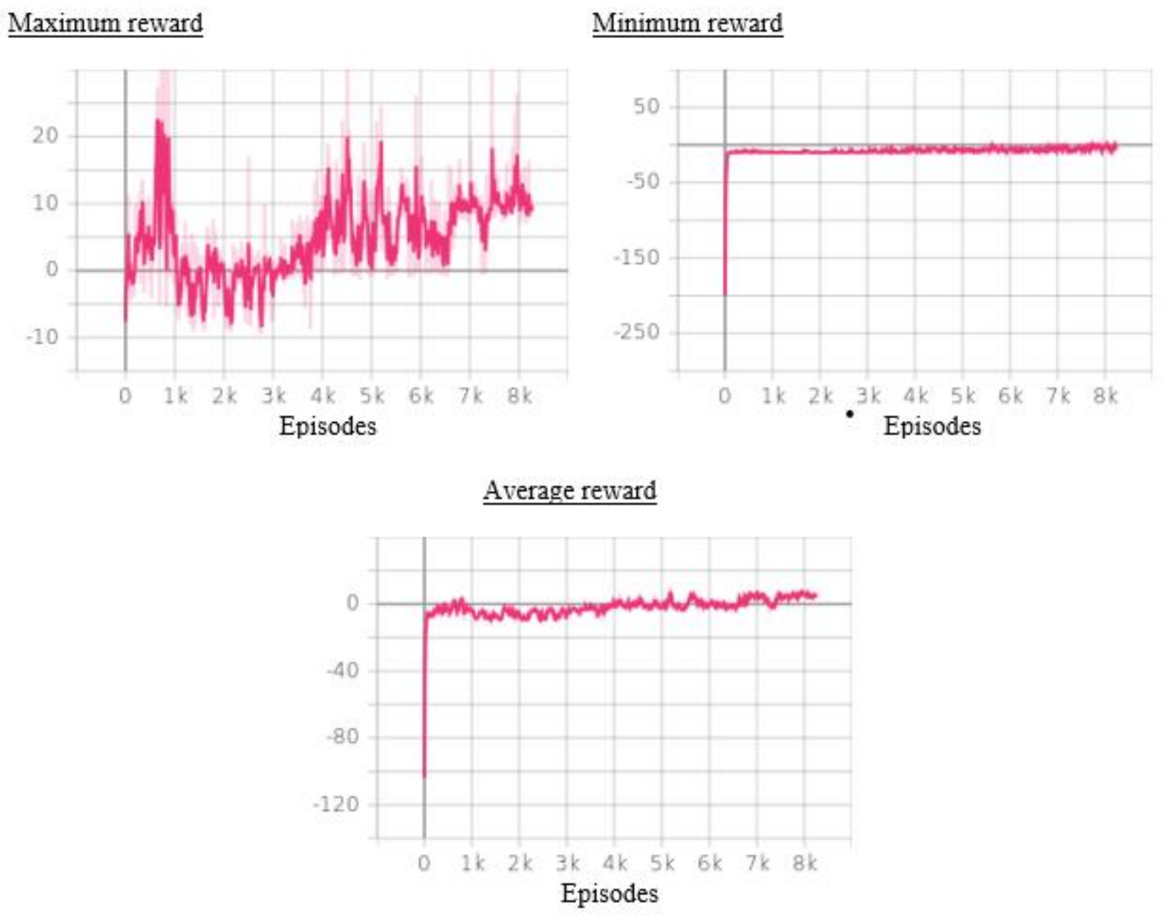


Figure 29: Rewards from Testing 2. Source: Own compilation

13. Conclusions and Future Work

13.1 Conclusions

In the early stages it was difficult to think that we would be able to do sufficient research on a topic like autonomous driving. In addition, inexperience, limited time, scarce resources, and constrained computational power meant that a good result would be rather challenging. Nevertheless, it was decided to go ahead.

As commented on the goals of the thesis, the main objective of this project is to make an autonomous car learn how to drive any given route. To accomplish this objective, it had to be divided into two parts: to set a base on autonomous driving with CARLA by creating and setting up an environment for CARLA and to teach an artificial intelligence how to drive a car with a Deep Q-Network algorithm.

This first objective was successfully achieved by creating a tutorial that explained how to replicate the development environment. The second objective was also attained as the agent was able to follow an established route while respecting basic traffic signs. This was thanks to the implementation of the waypoint and reward systems which contributed to the resolution of the problem.

Although these specific objectives have been met there is still continuity and a wide range of improvement.

13.2 Future Work

One of the advantages of this field is that there is always room for improvement in every aspect. In the following list you can find a couple of ideas on how our work can be broadened or improved:

- Supporting an installation tutorial for Linux or other operating systems can increase accessibility and encourage the creation of more projects in the CARLA simulation software.

- Using other machine learning technologies and techniques such as the Proximal Policy Optimizations instead of the Deep Q-Network algorithm would be interested to study in comparison.

- Changing the epsilon decay equation with an epsilon greedy strategy to improve the exploration-exploitation stages.

- Adding a HUD to the car as it will look better during the training session. It could show the reward obtained at that moment, the total distance travelled, the deviation between the car itself and the centre of the road and other important information.

14. References

[1] The 6 Levels of Vehicle Autonomy Explained | Synopsys Automotive. (s. f.). The 6 Levels of Vehicle Autonomy. Retrieved March 1, 2022, from https://www.synopsys.com/automotive/autonomous-driving-levels.html

[2] Self-driving cars in the EU: from science fiction to reality | News | European Parliament. (2019, January 14). Europarl. Retrieved March 21, 2022, from https://www.europarl.europa.eu/news/en/headlines/economy/20190110STO23102/self-driving-cars-in-the-eu-from-science-fiction-to-reality

[3] Machine Learning: What it is and why it matters. (s. f.). SAS. Retrieved March 1, 2022, from https://www.sas.com/en_us/insights/analytics/machine-learning.html

[4] Carew, J. M. (2021, 29 Marzo). Reinforcement Learning. SearchEnterpriseAI. Retrieved March 1, 2022, from https://www.techtarget.com/searchenterpriseai/definition/reinforcement-learning#:%7E:text=Reinforcement%20learning%20is%20a%20machine,learn%20through%20trial%20and%20error.

[5] Introduction to RL and Deep Q Networks | TensorFlow Agents. (n.d.). TensorFlow. Retrieved June 22, 2022, from https://www.tensorflow.org/agents/tutorials/0_intro_rl

[6] Education, I. C. (2021, 3 August). Neural Networks. Neural Networks. Retrieved March 1, 2022, from https://www.ibm.com/cloud/learn/neural-networks#:%7E:text=Neural%20networks%2C%20also%20known%20as,neurons%20signal%20to%20one%20another.

[7] Wikipedia contributors. (2022, March 1). Self-driving car. Wikipedia. Retrieved March 1, 2022, from https://en.wikipedia.org/wiki/Self-driving_car#:%7E:text=%20An%20autonomous%20vehicle%20may%20operate%20on%20a,of%20operating%20in%20compliance%20with%20the. . .%20More%20

[8] Development Team, C. A. R. L. A. (s. f.). CARLA. CARLA Simulator. Retrieved March 1, 2022, from https://carla.org/

[9] Puzhevich, V. (2021, October 27). Functional vs Non-Functional Requirements: The Definitive Guide. SCAND. Retrieved March 1, 2022, from https://scand.com/company/blog/functional-vs-non-functional-requirements/

[10] SigOpt. (2021b, June 16). Common Problems in Hyperparameter Optimization. Retrieved March 1, 2022, from https://sigopt.com/blog/common-problems-in-hyperparameter-optimization/

[11] Atlassian. (n.d.). What is Agile? Retrieved March 1, 2022, from https://www.atlassian.com/agile#:%7E:text=Agile%20is%20an%20iterative%20approach,small%2C%20but%20consumable%2C%20increments.

[12] Learn, N. (n.d.). The Agile Process 101: Understanding the Benefits of Using Agile Methodology. Nvisia. Retrieved March 21, 2022, from https://www.nvisia.com/insights/agile-methodology

[13] Anaconda | Anaconda Distribution. (n.d.). Anaconda. Retrieved June 22, 2022, from https://www.anaconda.com/products/distribution

[14] Hays. (n.d.). Hays Recruiting Experts Worldwide. Retrieved March 21, 2022, from https://www.hays.es/

[15] Tarifaluzhora.es. (n.d.). ¿Cuánto cuesta el kilovatio hora de luz (kWh) en España? Retrieved March 21, 2022, from https://tarifaluzhora.es/info/precio-kwh

[16] NVIDIA CUDA GPUs - Compute Capability. (2022, May 31). NVIDIA Developer. Retrieved June 22, 2022, from https://developer.nvidia.com/cuda-gpus

[17] Wikipedia contributors. (2022a, June 15). Visual Studio Code. Wikipedia. Retrieved June 22, 2022, from https://en.wikipedia.org/wiki/Visual_Studio_Code

[18] C. (n.d.). Releases · carla-simulator/carla. GitHub. Retrieved June 22, 2022, from https://github.com/carla-simulator/carla/releases

[19] Marta B. (2020, January 20) Study of the implementation of an autonomous driving system. Bachelor Thesis. Retrieve June 22, 2022, from https://upcommons.upc.edu/bitstream/handle/2117/180817/TFG_Marta_Basquens.pdf

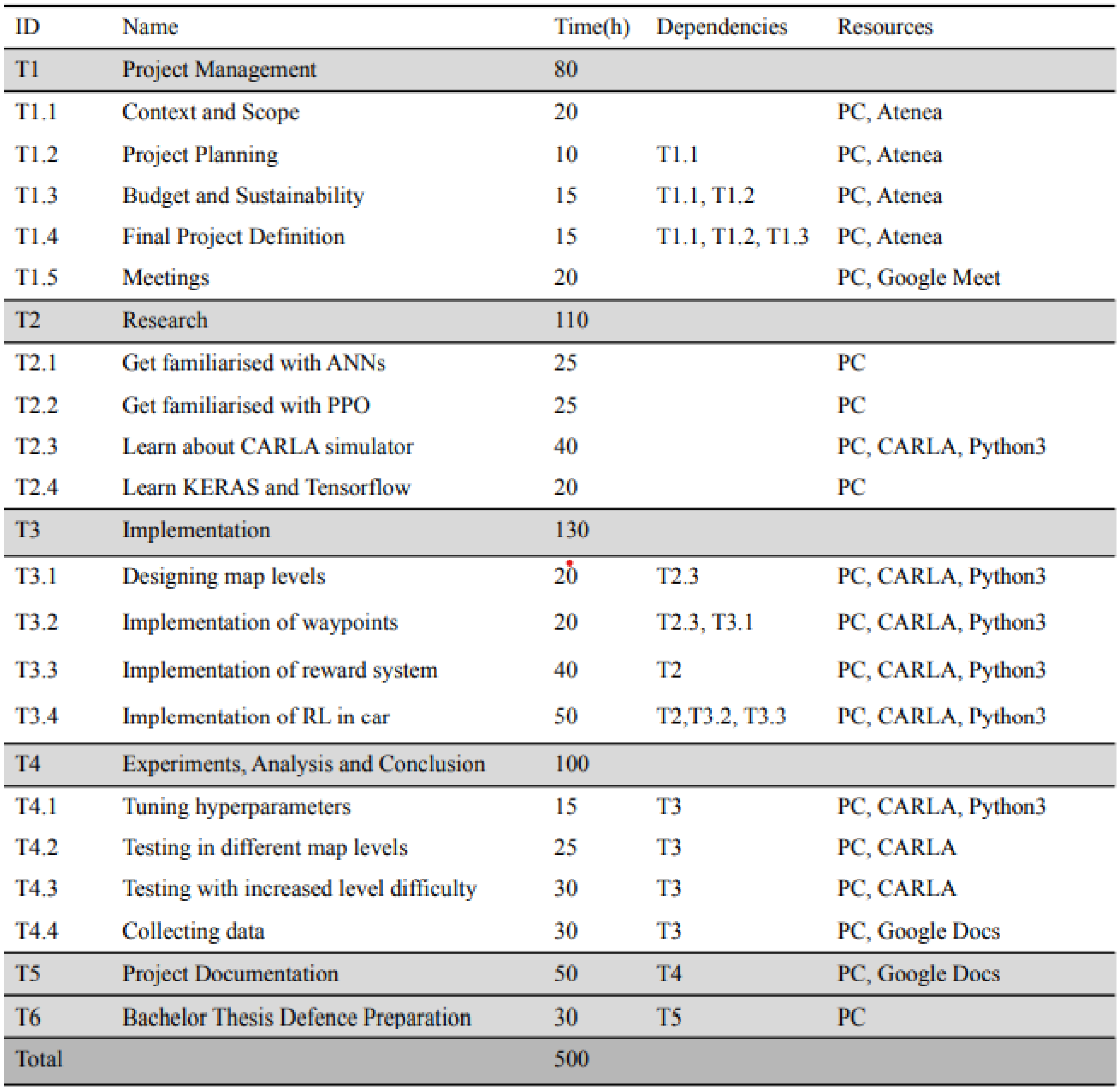
[20] M. (n.d.-b). Exploration vs. Exploitation in Reinforcement Learning. Manifold. Retrieved June 22, 2022, from https://www.manifold.ai/exploration-vs-exploitation-in-reinforcement-learning

[21] Kendall, A. (2018, July 1). Learning to Drive in a Day. arXiv.Org. Retrieved June 22, 2022, from https://arxiv.org/abs/1807.00412

[22] Raffin, A. (2022, June 5). Learning to Drive Smoothly in Minutes - Towards Data Science. Medium. Retrieved June 22, 2022, from https://towardsdatascience.com/learning-to-drive-smoothly-in-minutes-450a7cdb35f4

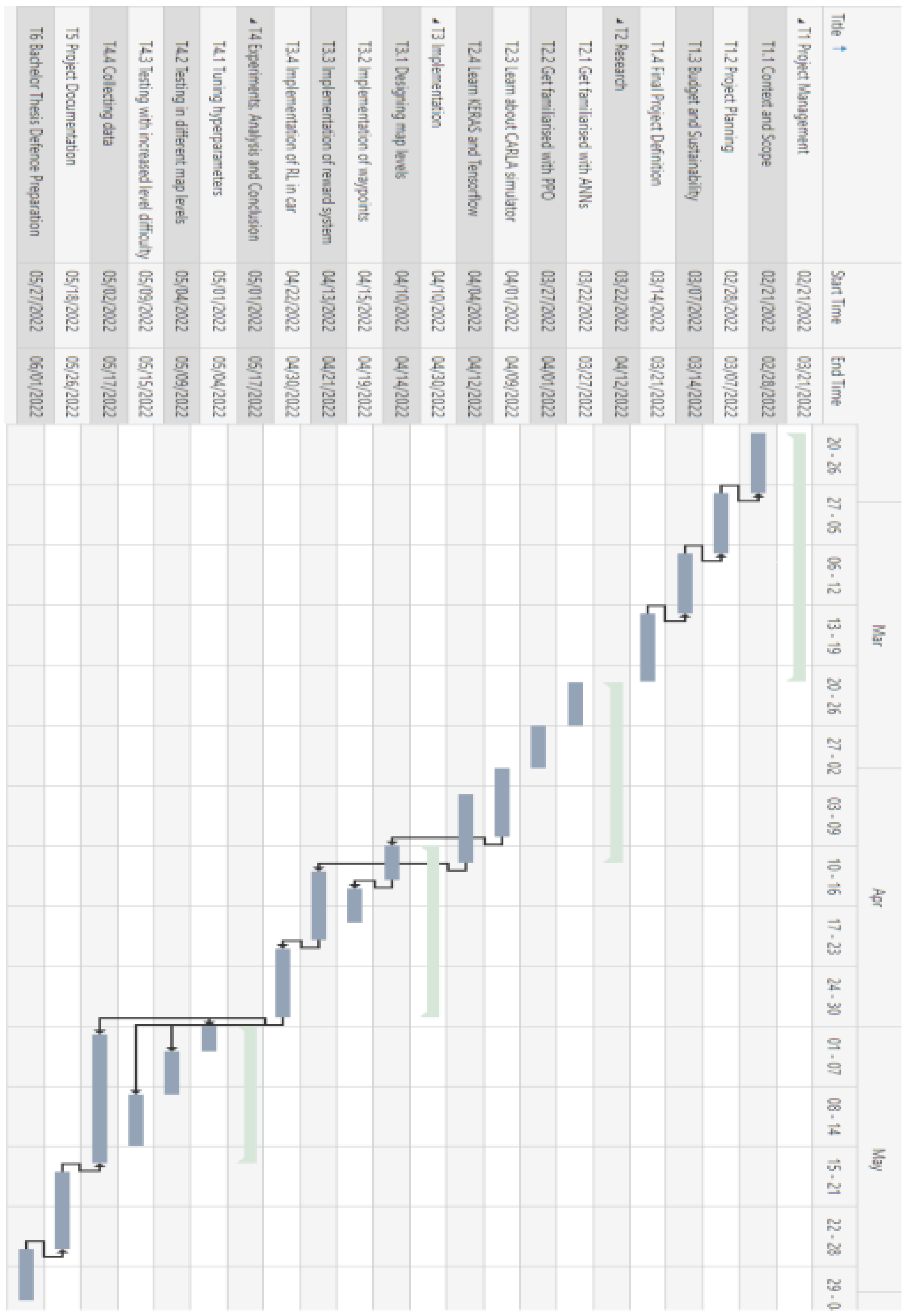
A. Appendix

A.1 Summary of the Tasks (First Version)



ID	Name	Time(h)	Dependencies	Resources
T1	Project Management	80		
T1.1	Context and Scope	20		PC, Atenea
T1.2	Project Planning	10	T1.1	PC, Atenea
T1.3	Budget and Sustainability	15	T1.1, T1.2	PC, Atenea
T1.4	Final Project Definition	15	T1.1, T1.2, T1.3	PC, Atenea
T1.5	Meetings	20		PC, Google Meet
T2	Research	110		
T2.1	Get familiarised with ANNs	25		PC
T2.2	Get familiarised with PPO	25		PC
T2.3	Learn about CARLA simulator	40		PC, CARLA, Python3
T2.4	Learn KERAS and Tensorflow	20		PC
T3	Implementation	130		
T3.1	Designing map levels	20	T2.3	PC, CARLA, Python3
T3.2	Implementation of waypoints	20	T2.3, T3.1	PC, CARLA, Python3
T3.3	Implementation of reward system	40	T2	PC, CARLA, Python3
T3.4	Implementation of RL in car	50	T2,T3.2, T3.3	PC, CARLA, Python3
T4	Experiments, Analysis and Conclusion	100		
T4.1	Tuning hyperparameters	15	T3	PC, CARLA, Python3
T4.2	Testing in different map levels	25	T3	PC, CARLA
T4.3	Testing with increased level difficulty	30	T3	PC, CARLA
T4.4	Collecting data	30	T3	PC, Google Docs
T5	Project Documentation	50	T4	PC, Google Docs
T6	Bachelor Thesis Defence Preparation	30	T5	PC
Total		500		

A.2 Gantt Chart (First Version)

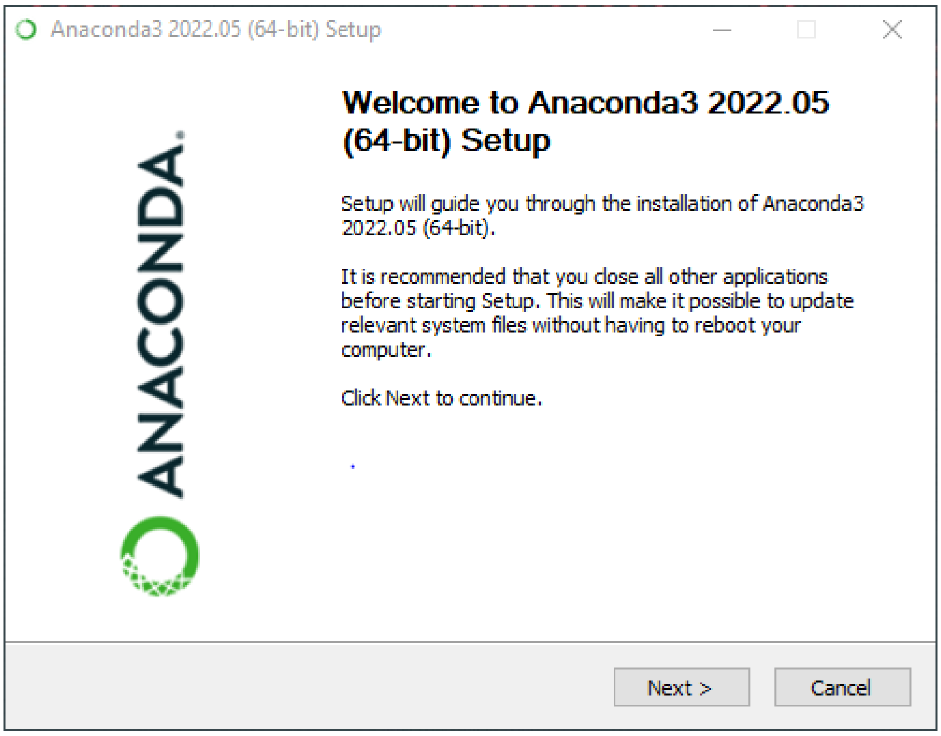


Title ↑	Start Time	End Time
◢ T1 Project Management	02/21/2022	03/21/2022
T1.1 Context and Scope	02/21/2022	02/28/2022
T1.2 Project Planning	02/28/2022	03/07/2022
T1.3 Budget and Sustainability	03/07/2022	03/14/2022
T1.4 Final Project Definition	03/14/2022	03/21/2022
◢ T2 Research	03/22/2022	04/12/2022
T2.1 Get familiarised with ANNs	03/22/2022	03/27/2022
T2.2 Get familiarised with PPO	03/27/2022	04/01/2022
T2.3 Learn about CARLA simulator	04/01/2022	04/09/2022
T2.4 Learn KERAS and Tensorflow	04/04/2022	04/12/2022
◢ T3 Implementation	04/10/2022	04/30/2022
T3.1 Designing map levels	04/10/2022	04/14/2022
T3.2 Implementation of waypoints	04/15/2022	04/19/2022
T3.3 Implementation of reward system	04/13/2022	04/21/2022
T3.4 Implementation of RL in car	04/22/2022	04/30/2022
◢ T4 Experiments, Analysis and Conclusion	05/01/2022	05/17/2022
T4.1 Tuning hyperparameters	05/01/2022	05/04/2022
T4.2 Testing in different map levels	05/04/2022	05/09/2022
T4.3 Testing with increased level difficulty	05/09/2022	05/15/2022
T4.4 Collecting data	05/02/2022	05/17/2022
T5 Project Documentation	05/18/2022	05/26/2022
T6 Bachelor Thesis Defence Preparation	05/27/2022	06/01/2022

B. Appendix

B.1 Anaconda Installation Guide

Step 1



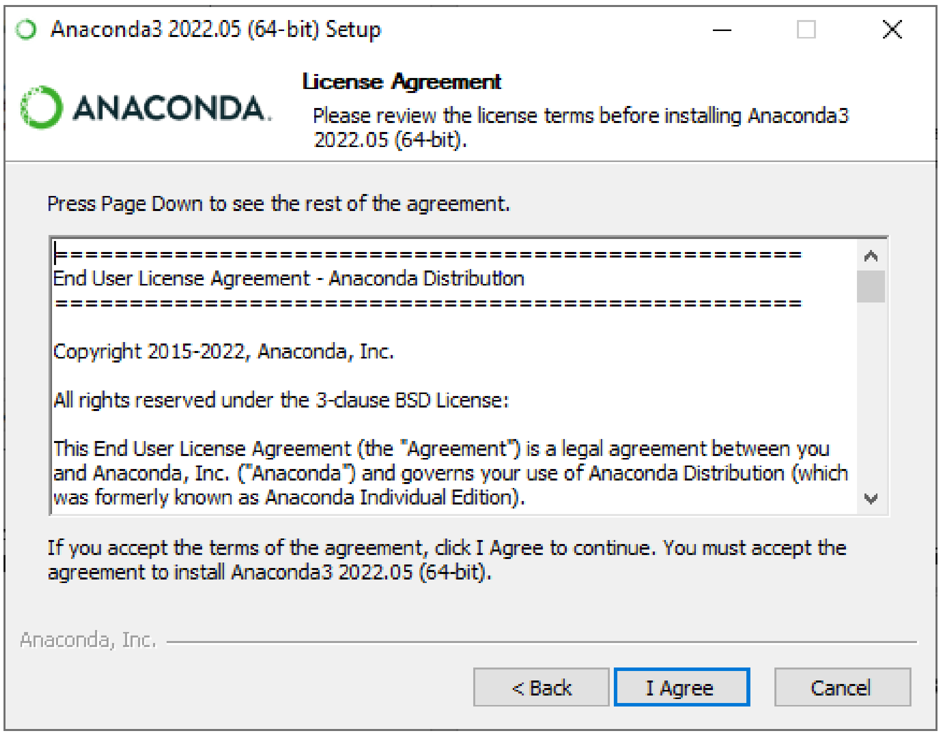
Welcome to Anaconda3 2022.05 (64-bit) Setup

Setup will guide you through the installation of Anaconda3 2022.05 (64-bit).

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Step 2



License Agreement

Please review the license terms before installing Anaconda3 2022.05 (64-bit).

Press Page Down to see the rest of the agreement.

```
==================================================
End User License Agreement - Anaconda Distribution
==================================================
```

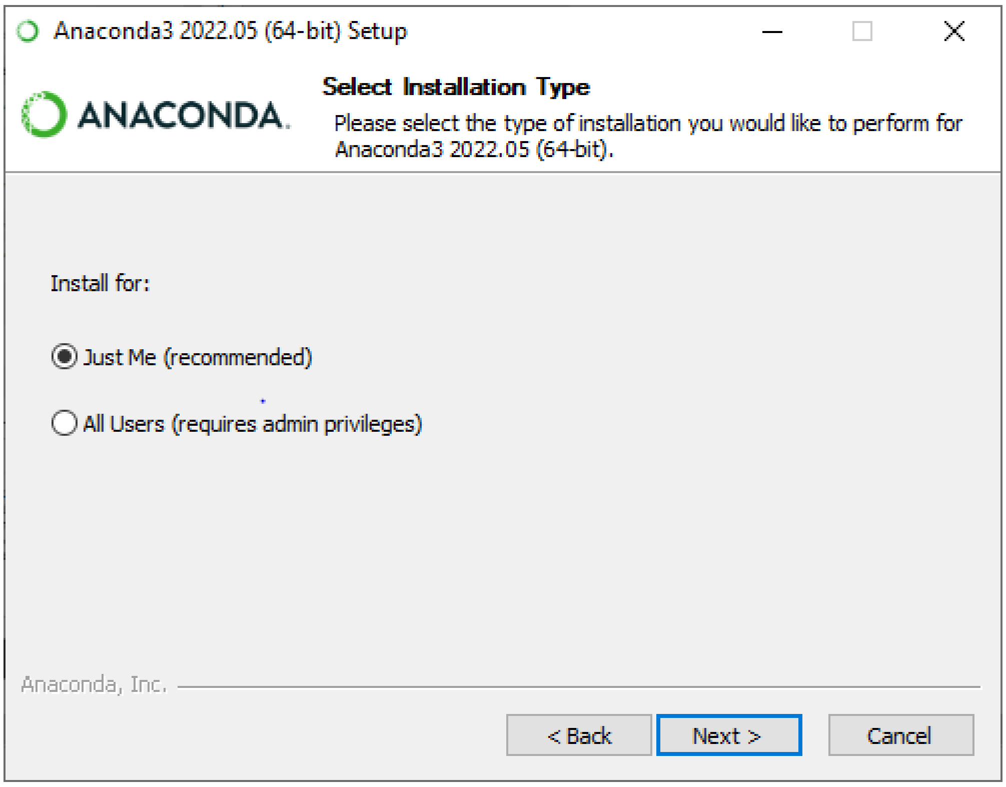
Copyright 2015-2022, Anaconda, Inc.

All rights reserved under the 3-clause BSD License:

This End User License Agreement (the "Agreement") is a legal agreement between you and Anaconda, Inc. ("Anaconda") and governs your use of Anaconda Distribution (which was formerly known as Anaconda Individual Edition).

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Anaconda3 2022.05 (64-bit).

Step 3


Anaconda3 2022.05 (64-bit) Setup

Select Installation Type

Please select the type of installation you would like to perform for Anaconda3 2022.05 (64-bit).

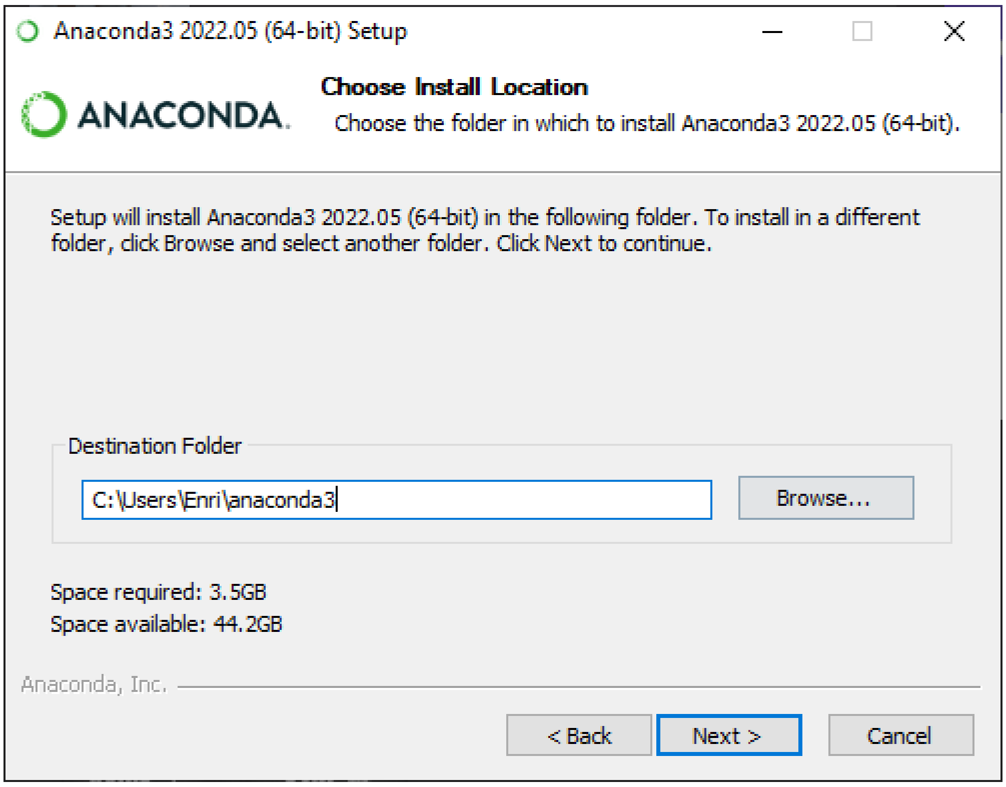
Install for:

[✓ Just Me (recommended)]

All Users (requires admin privileges)

Anaconda, Inc.

Step 4


Anaconda3 2022.05 (64-bit) Setup

Choose Install Location

Choose the folder in which to install Anaconda3 2022.05 (64-bit).

Setup will install Anaconda3 2022.05 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

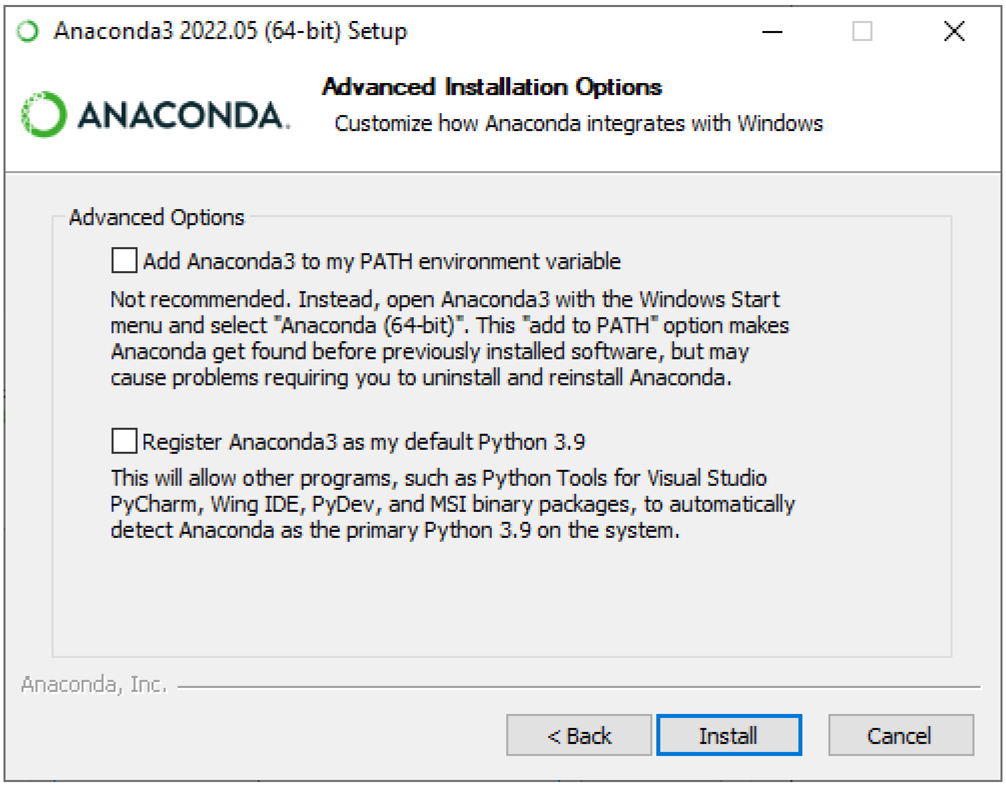
Destination Folder

C:\Users\Enri\anaconda3

Space required: 3.5GB
Space available: 44.2GB

Anaconda, Inc.

Step 5



Advanced Installation Options

Customize how Anaconda integrates with Windows

Advanced Options

☐ Add Anaconda3 to my PATH environment variable

Not recommended. Instead, open Anaconda3 with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.

☐ Register Anaconda3 as my default Python 3.9

This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.9 on the system.

B.2 Library List

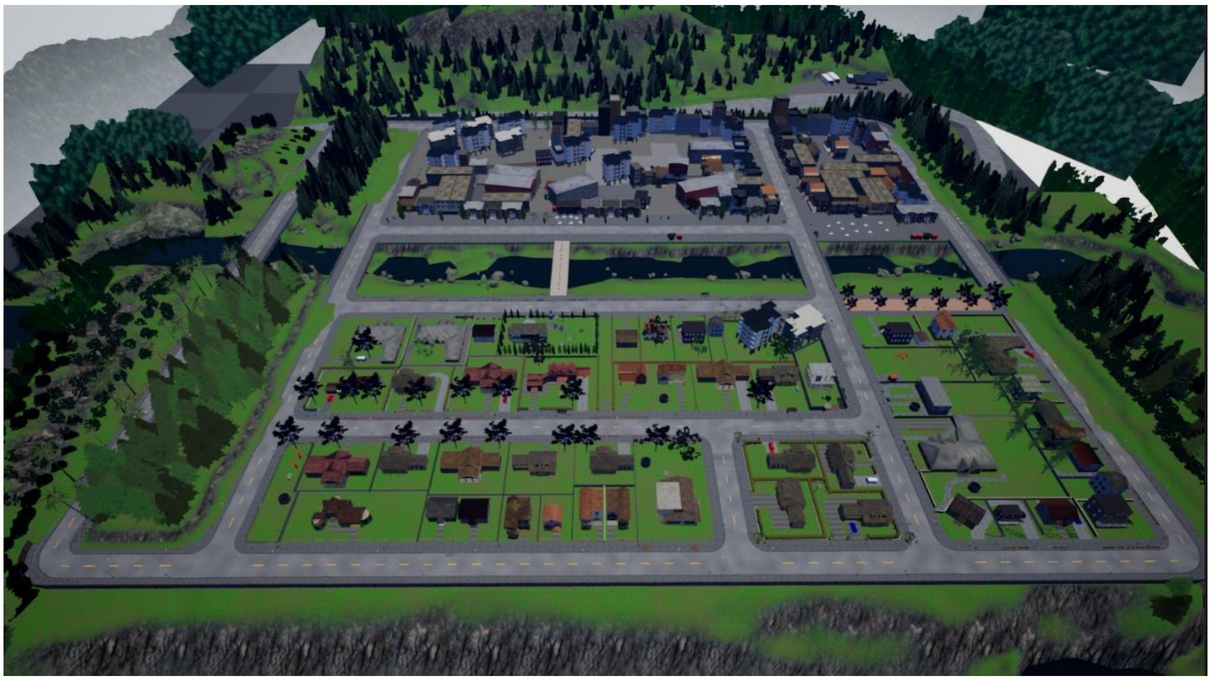
```
# packages in environment at C:\Users\Enri\anaconda3\envs\carlaEnv:
#
# Name                    Version                   Build  Channel
_tflow_select             2.1.0                       gpu
absl-py                   0.15.0             pyhd3eb1b0_0
astor                     0.8.1            py37haa95532_0
blas                      1.0                         mkl
ca-certificates           2022.4.26            haa95532_0
cached-property           1.5.2                      py_0
certifi                   2022.5.18.1      py37haa95532_0
colorama                  0.4.4              pyhd3eb1b0_0
cudatoolkit               10.0.130                      0
cudnn                     7.6.5                cuda10.0_0
dataclasses               0.8                pyh6d0b6a4_7
freeglut                  3.2.2                h0e60522_1    conda-forge
freetype                  2.10.4               h546665d_1    conda-forge
gast                      0.5.3              pyhd3eb1b0_0
grpcio                    1.42.0           py37hc60d5dd_0
h5py                      3.6.0            py37h3de5c98_0
hdf5                      1.10.6               h7ebc959_0
icc_rt                    2019.0.0             h0cc432a_1
icu                       69.1                 h0e60522_0    conda-forge
importlib-metadata        4.11.3           py37haa95532_0
intel-openmp              2021.4.0          haa95532_3556
jasper                    2.0.33               h77af90b_0    conda-forge
jpeg                      9e                   h8ffe710_1    conda-forge
keras                     2.2.4                         0
keras-applications        1.0.8                      py_1
keras-base                2.2.4                    py37_0
keras-preprocessing       1.1.2              pyhd3eb1b0_0
lerc                      3.0                  h0e60522_0    conda-forge
libblas                   3.9.0           1_h8933c1f_netlib    conda-forge
libcblas                  3.9.0           5_hd5c7e75_netlib    conda-forge
libclang                  13.0.1          default_h81446c8_0    conda-forge
libdeflate                1.10                 h8ffe710_0    conda-forge
liblapack                 3.9.0           5_hd5c7e75_netlib    conda-forge
liblapacke                3.9.0           5_hd5c7e75_netlib    conda-forge
libopencv                 4.5.5            py37h5f7ba43_9    conda-forge
libpng                    1.6.37               h1d00b33_2    conda-forge
libprotobuf               3.20.1               h23ce68f_0
libtiff                   4.3.0                hc4061b1_4    conda-forge
libwebp-base              1.2.2                h8ffe710_1    conda-forge
libzlib                   1.2.12               h8ffe710_0    conda-forge
lz4-c                     1.9.3                h8ffe710_1    conda-forge
m2w64-gcc-libgfortran     5.3.0                         6    conda-forge
m2w64-gcc-libs            5.3.0                         7    conda-forge
m2w64-gcc-libs-core       5.3.0                         7    conda-forge
m2w64-gmp                 6.1.0                         2    conda-forge
m2w64-libwinpthread-git   5.0.0.4634.697f757               2    conda-forge
markdown                  3.3.4            py37haa95532_0
mkl                       2021.4.0           haa95532_640
mkl-service               2.4.0            py37h2bbff1b_0
mkl_fft                   1.3.1            py37h277e83a_0
mkl_random                1.2.2            py37hf11a4ad_0
msys2-conda-epoch         20160418                      1    conda-forge
numpy                     1.21.5           py37h7a0a035_2
numpy-base                1.21.5           py37hca35cd5_2
```

```
opencv                    4.5.5            py37h03978a9_9    conda-forge
openssl                   1.1.1o               h2bbff1b_0
pip                       21.2.4           py37haa95532_0
protobuf                  3.20.1           py37hd77b12b_0
py-opencv                 4.5.5            py37h90c5f73_9    conda-forge
pyreadline                2.1                      py37_1
python                    3.7.13               h6244533_0
python_abi                3.7                     2_cp37m    conda-forge
pyyaml                    6.0              py37h2bbff1b_1
qt                        5.12.9               h556501e_6    conda-forge
scipy                     1.7.3            py37h0a974cb_0
setuptools                61.2.0           py37haa95532_0
six                       1.16.0             pyhd3eb1b0_1
sqlite                    3.38.3               h2bbff1b_0
tensorboard               1.14.0           py37he3c9ec2_0
tensorflow                1.14.0        gpu_py37h5512b17_0
tensorflow-base           1.14.0        gpu_py37h55fc52a_0
tensorflow-estimator      1.14.0                     py_0
tensorflow-gpu            1.14.0               h0d30ee6_0
termcolor                 1.1.0            py37haa95532_1
tqdm                      4.64.0           py37haa95532_0
typing_extensions         4.1.1              pyh06a4308_0
vc                        14.2                 h21ff451_1
vs2015_runtime            14.27.29016          h5e58377_2
werkzeug                  2.0.3              pyhd3eb1b0_0
wheel                     0.37.1             pyhd3eb1b0_0
wincertstore              0.2              py37haa95532_2
wrapt                     1.13.3           py37h2bbff1b_2
xz                        5.2.5                h62dcd97_1    conda-forge
yaml                      0.2.5                he774522_0
zipp                      3.8.0            py37haa95532_0
zlib                      1.2.12               h8ffe710_0    conda-forge
zstd                      1.5.2                h6255e5f_1    conda-forge
```

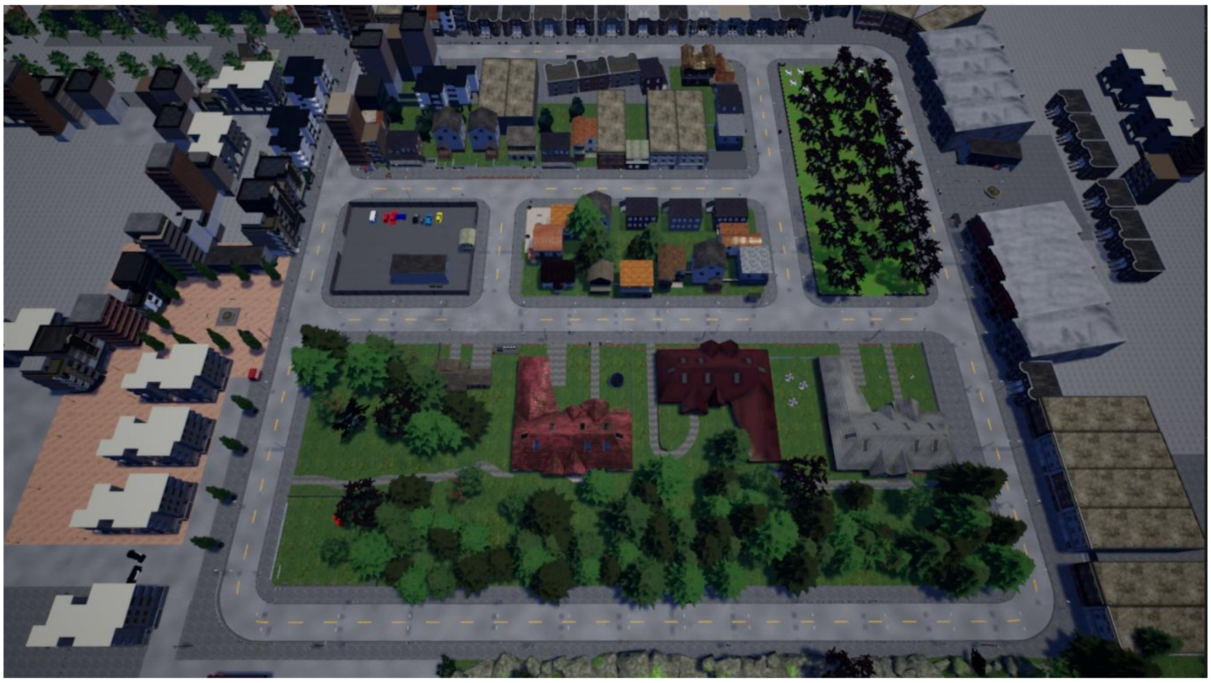
C. Appendix

C.1 Towns

TOWN01

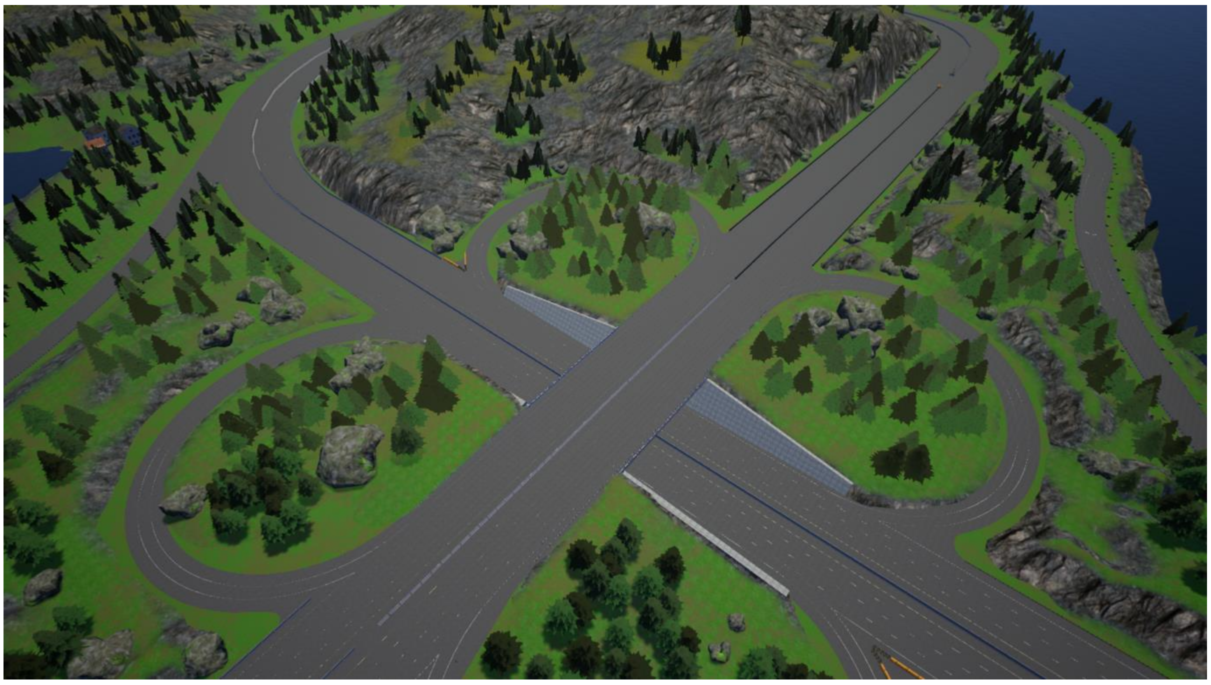


TOWN02



TOWN03

TOWN04



TOWN05

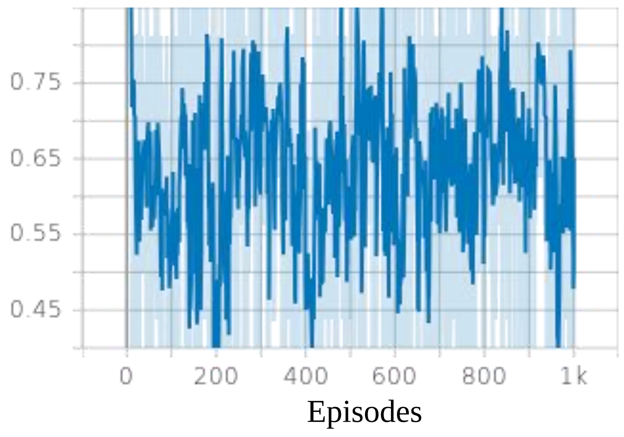
TOWN06

TOWN07

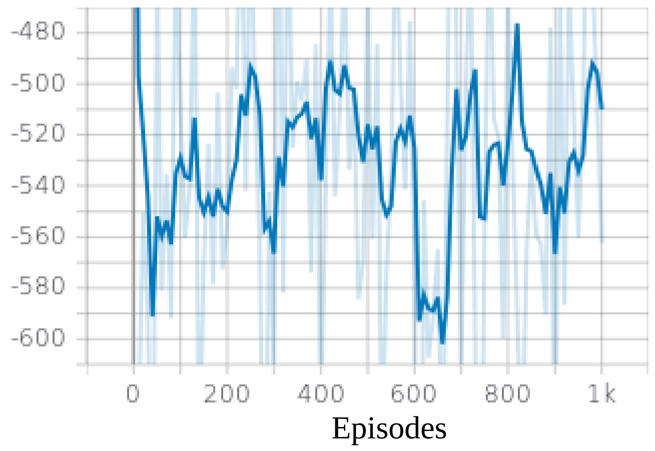
D. Appendix

D.1 Training Case 1

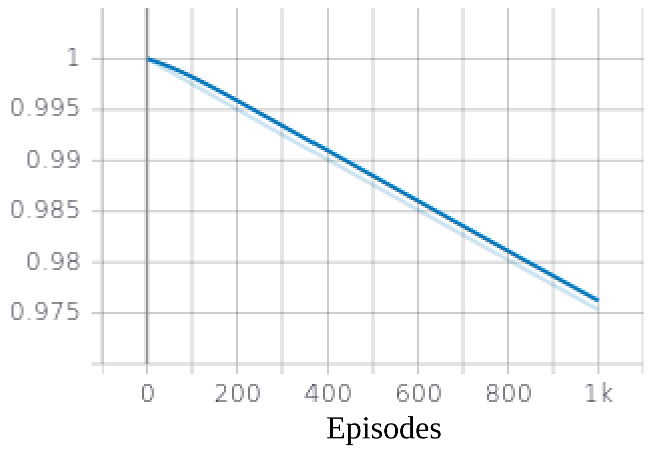
Accuracy



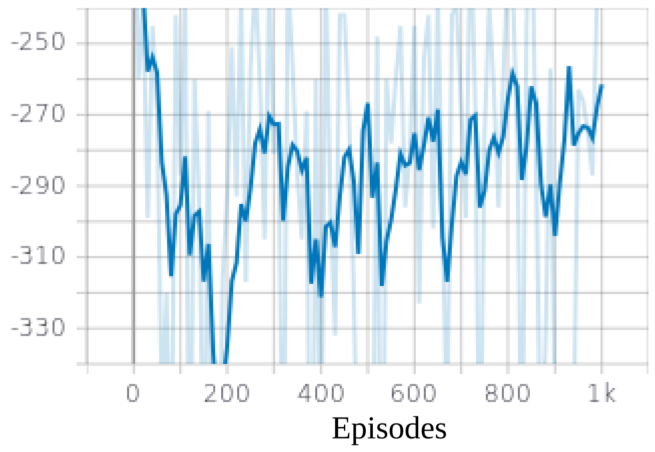
Average reward



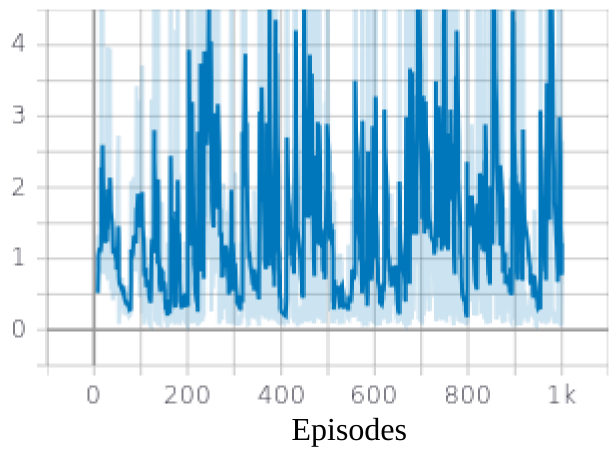
Epsilon decay



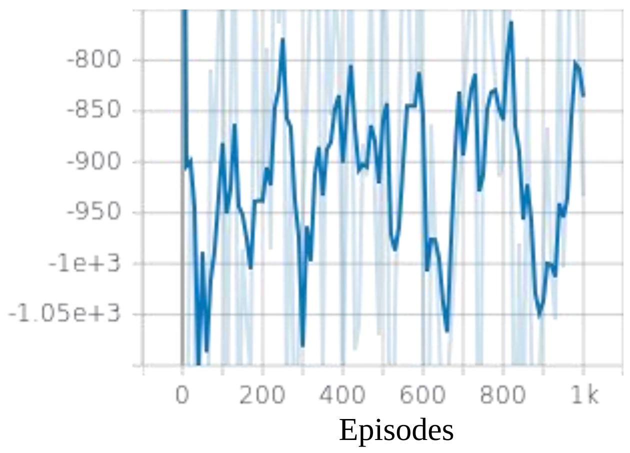
Maximum reward



Loss

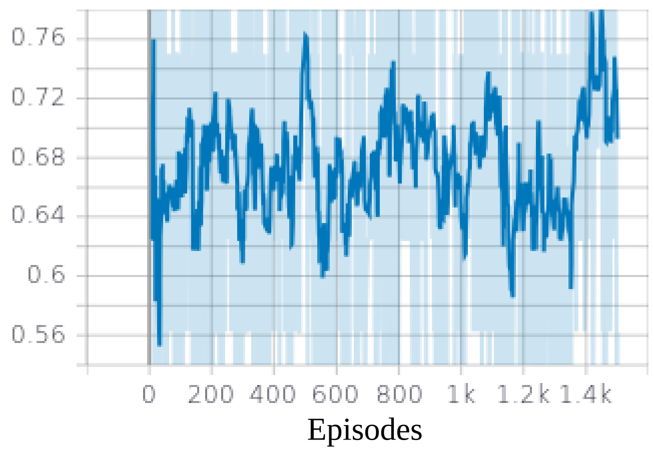


Minimum reward

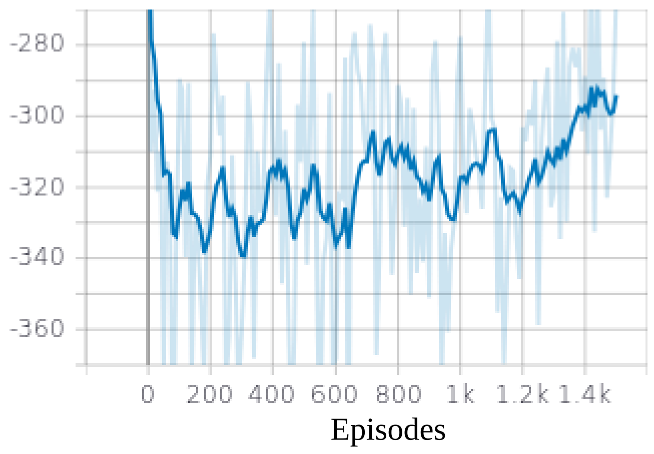


D.2 Training Case 2

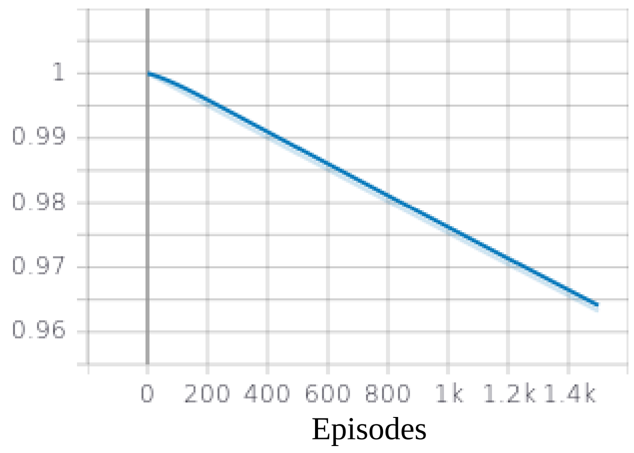
Accuracy



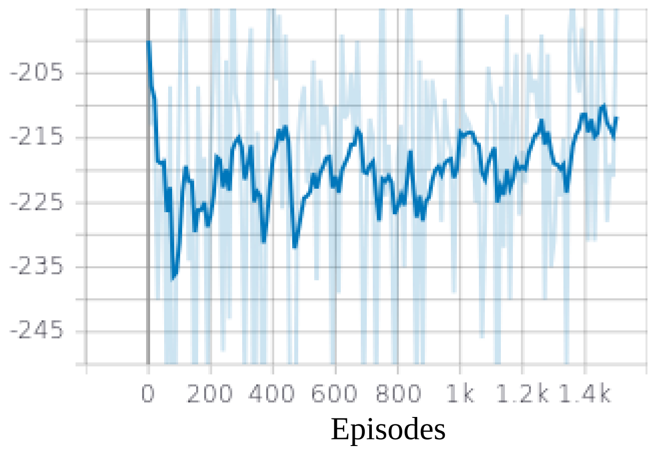
Average reward



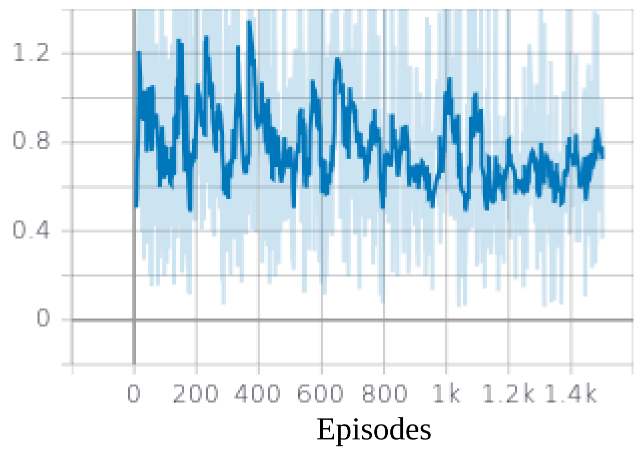
Epsilon decay



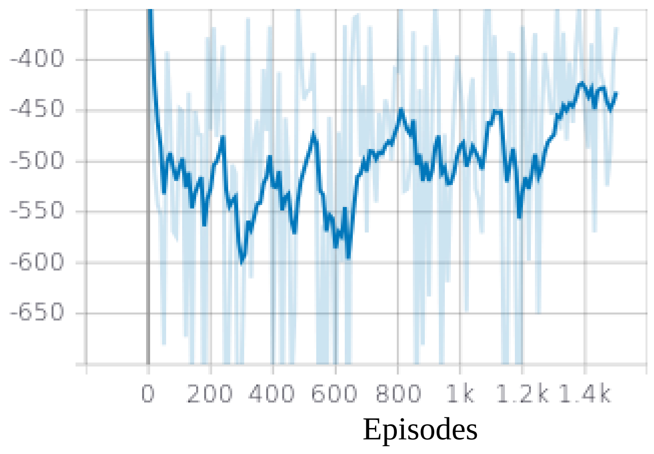
Maximum reward



Loss

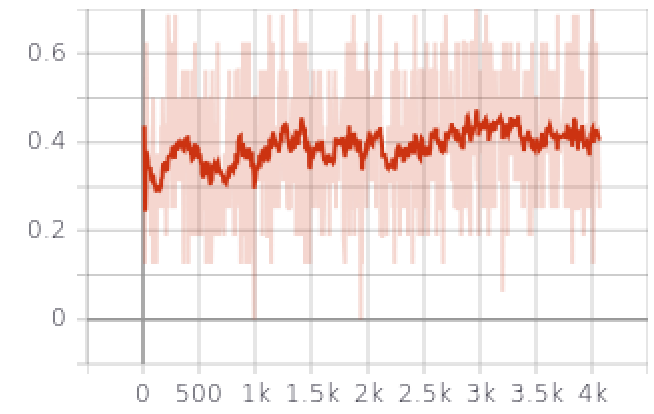


Minimum reward



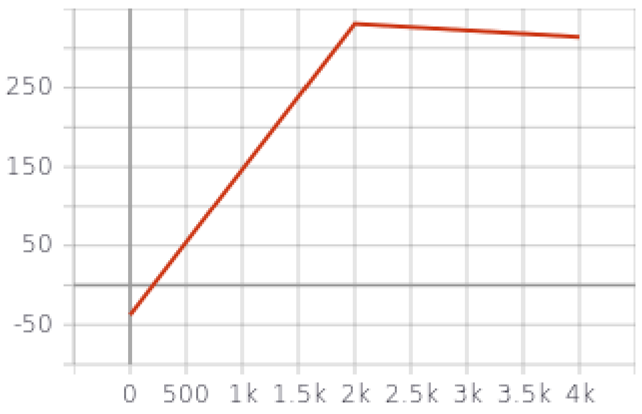
D.3 Training Case 3

Accuracy



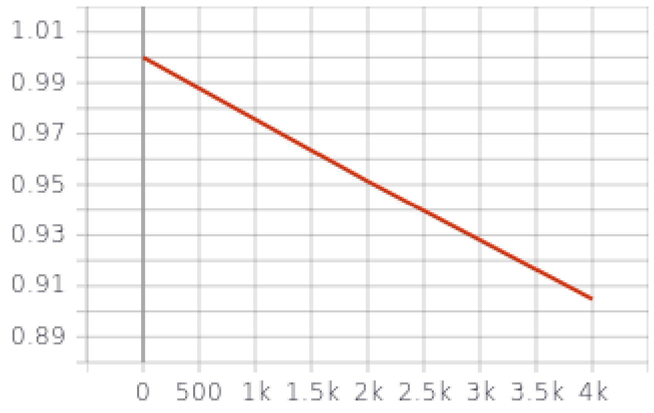
Episodes

Average reward



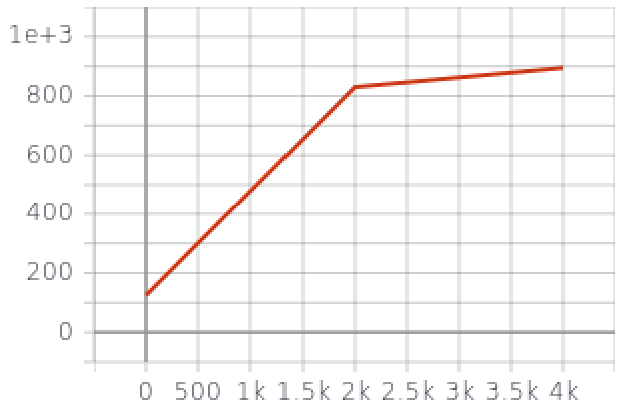
Episodes

Epsilon decay



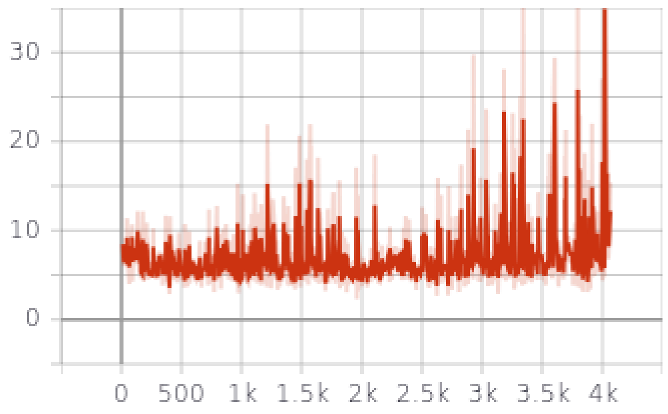
Episodes

Maximum reward



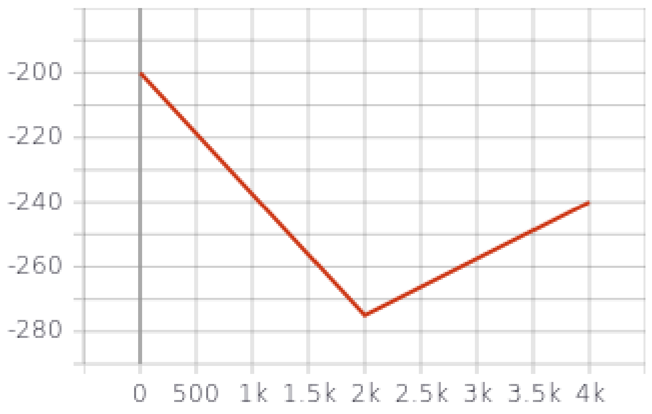
Episodes

Loss



Episodes

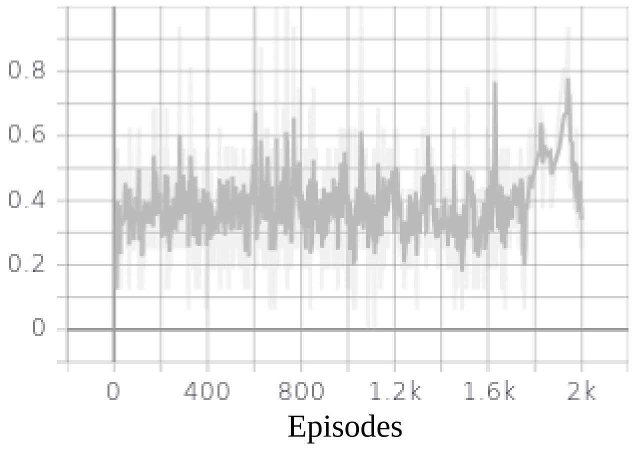
Minimum reward



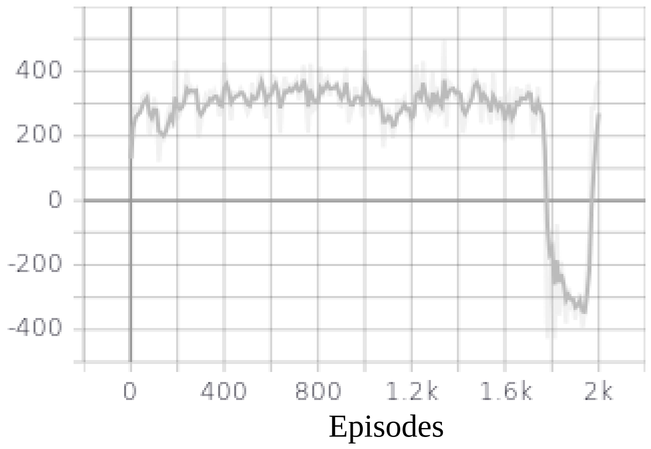
Episodes

D.4 Training Case 4

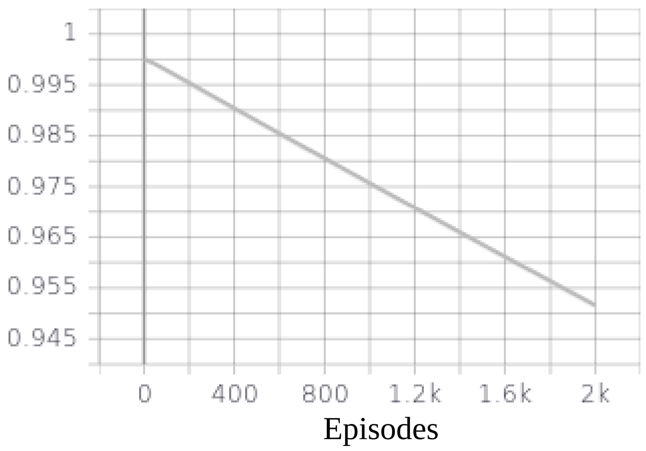
Accuracy



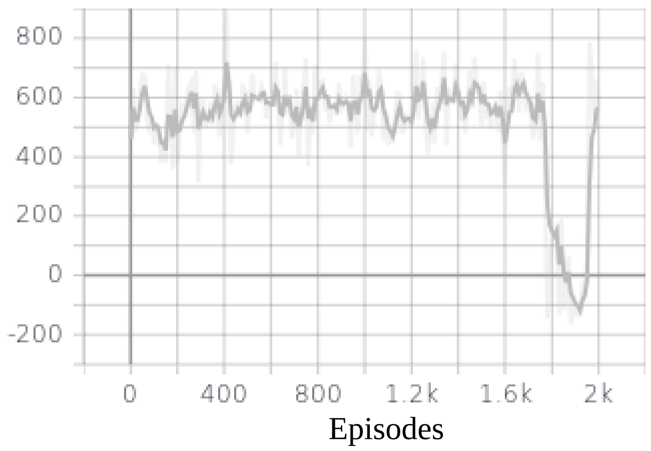
Average reward



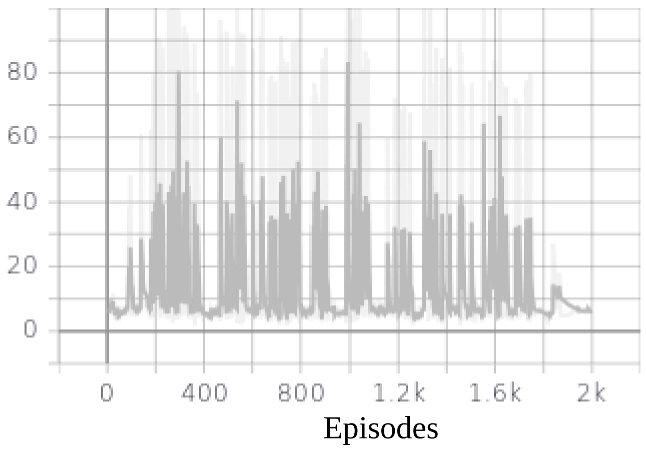
Epsilon decay



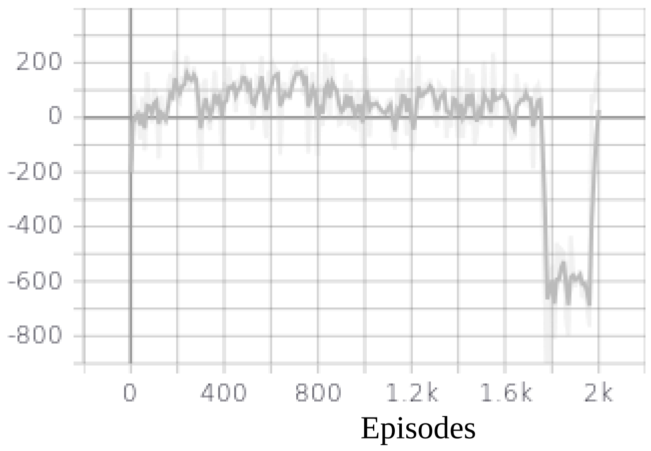
Maximum reward



Loss

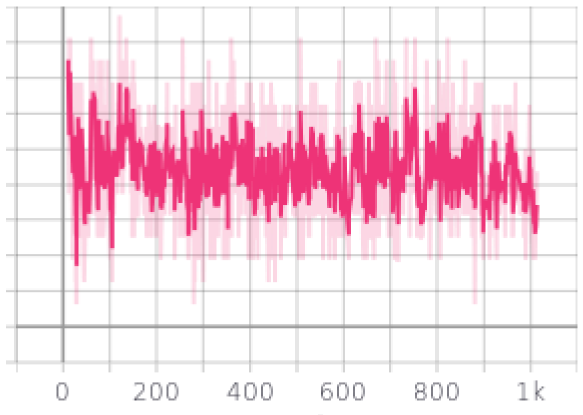


Minimum reward



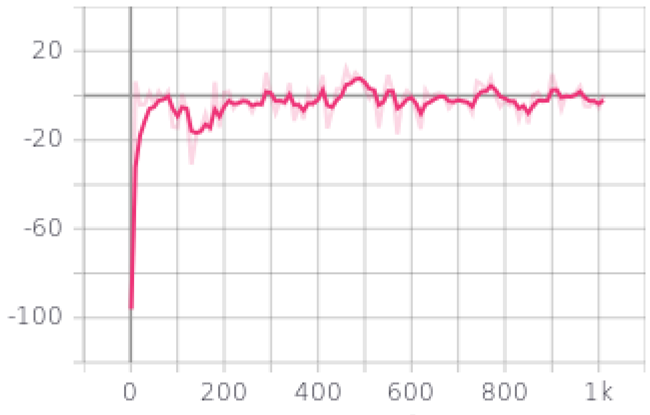
D.5 Training Case 5

Accuracy



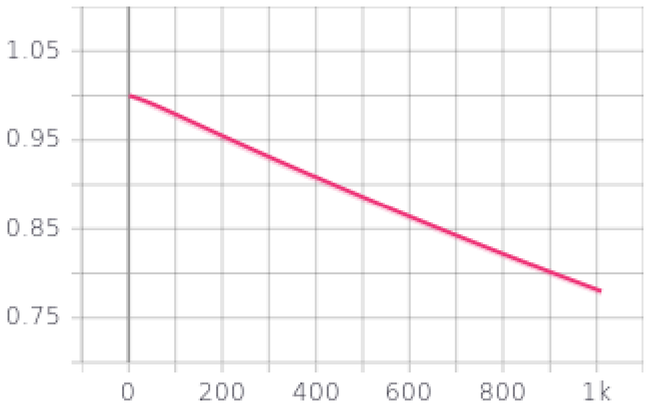
Episodes

Average reward



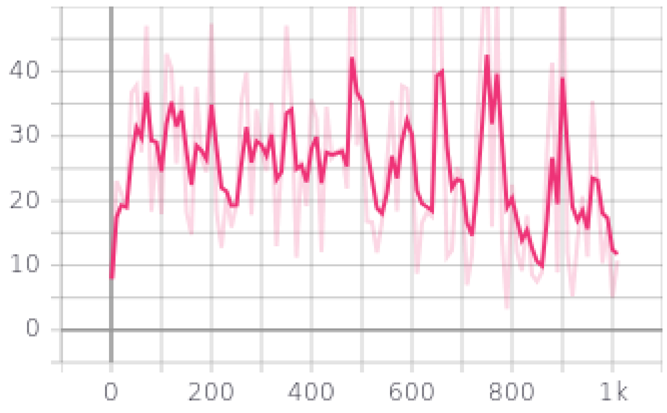
Episodes

Epsilon decay



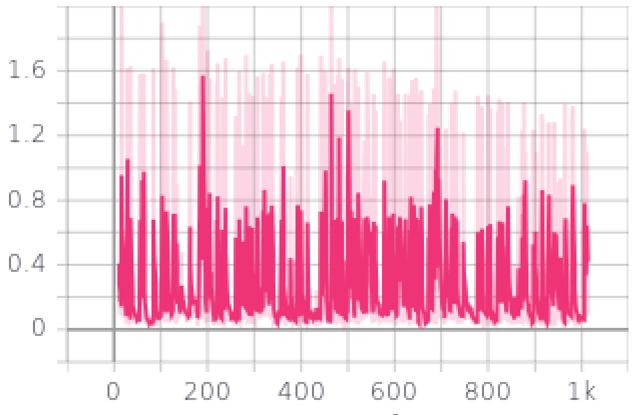
Episodes

Maximum reward



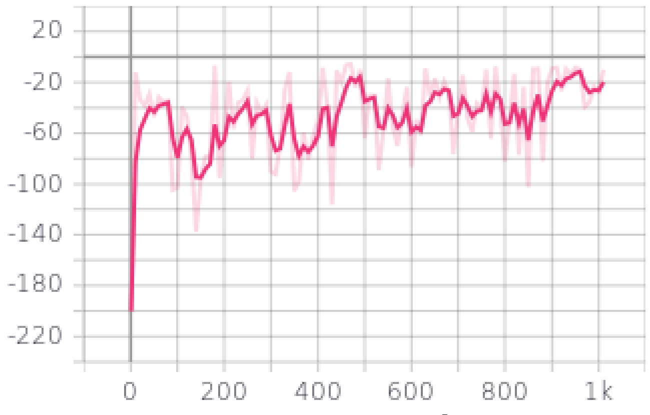
Episodes

Loss



Episodes

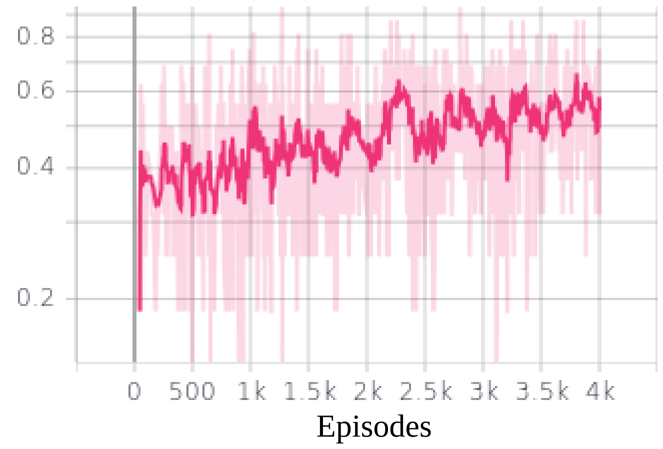
Minimum reward



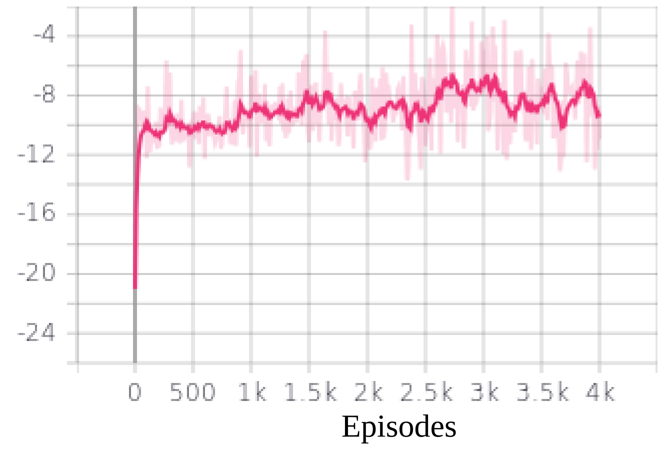
Episodes

D.6 Training Case 6

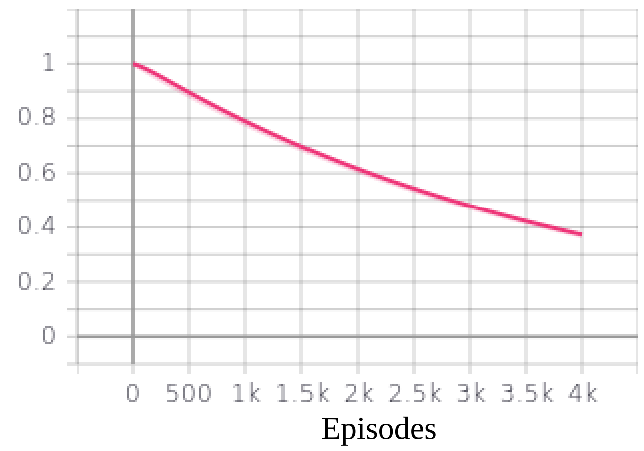
Accuracy



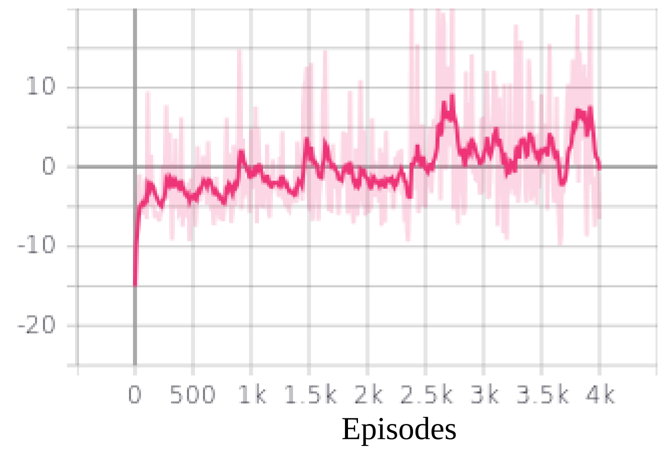
Average reward



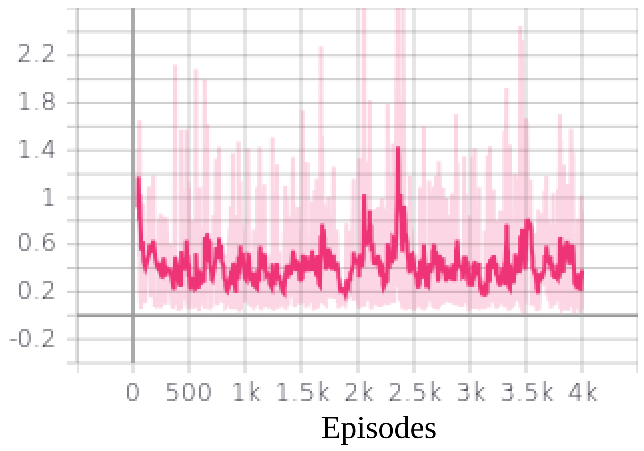
Epsilon decay



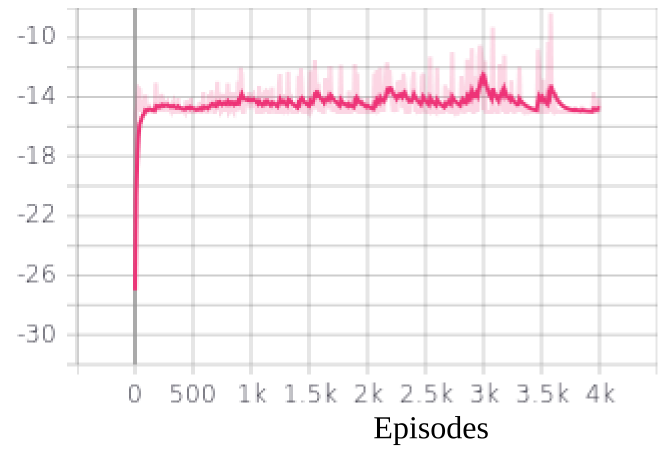
Maximum reward



Loss

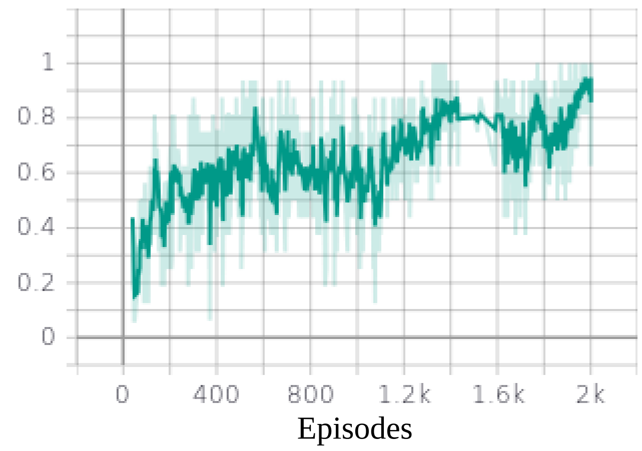


Minimum reward

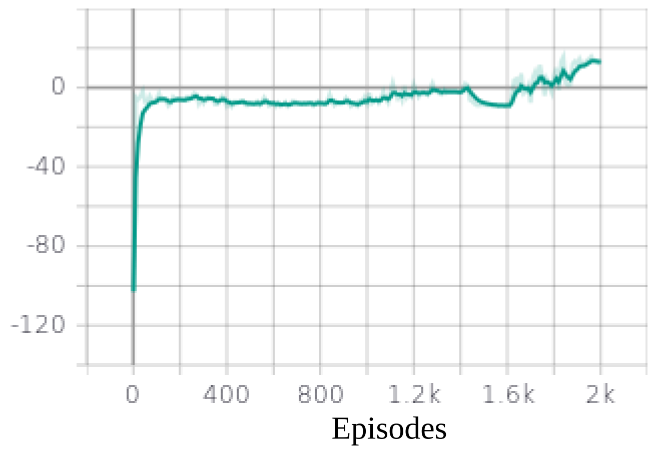


D.7 Training Case 7

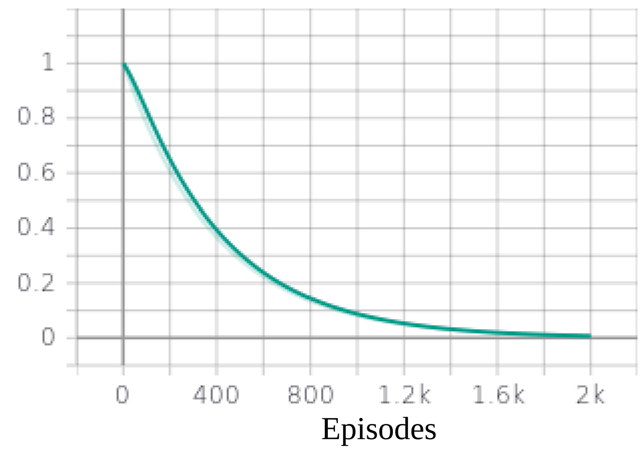
Accuracy



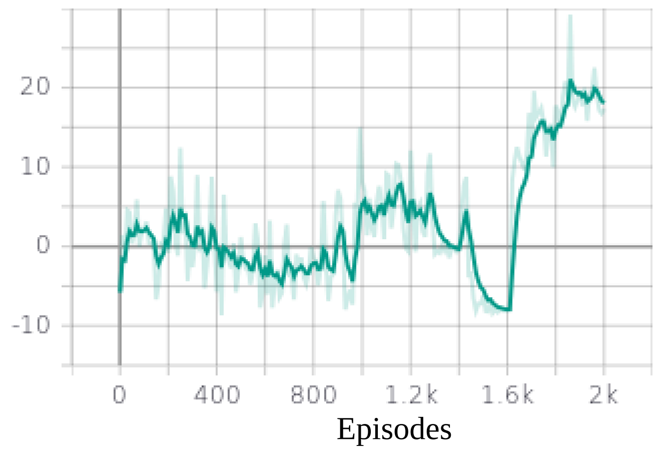
Average reward



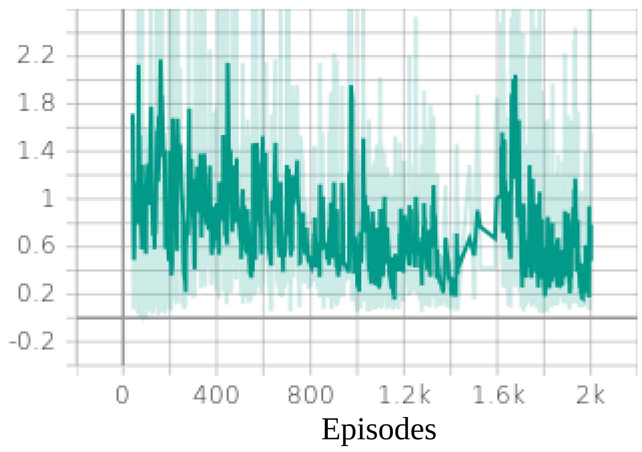
Epsilon decay



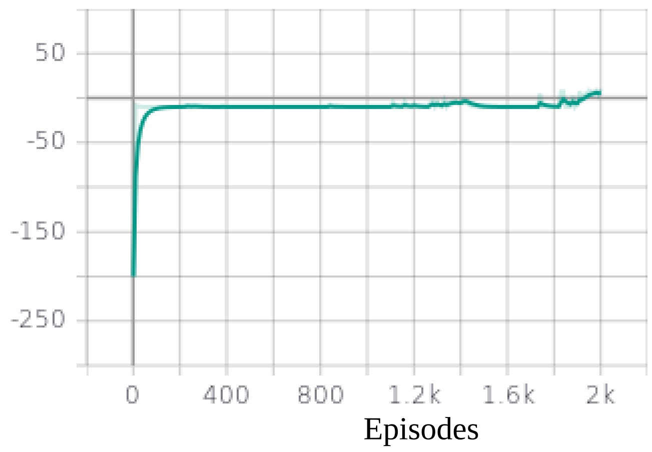
Maximum reward



Loss

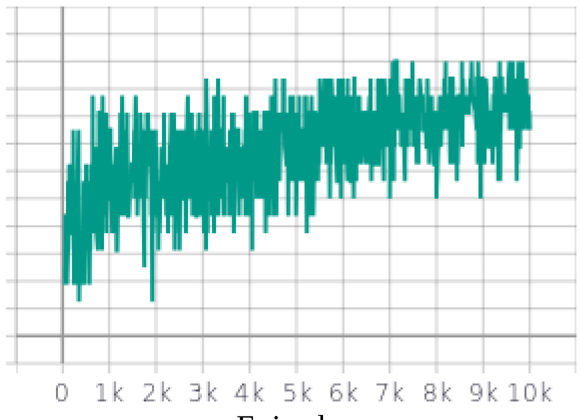


Minimum reward



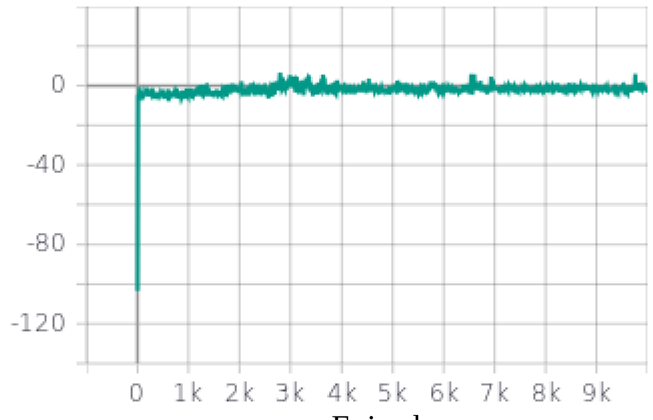
D.8 Training Case 8

Accuracy



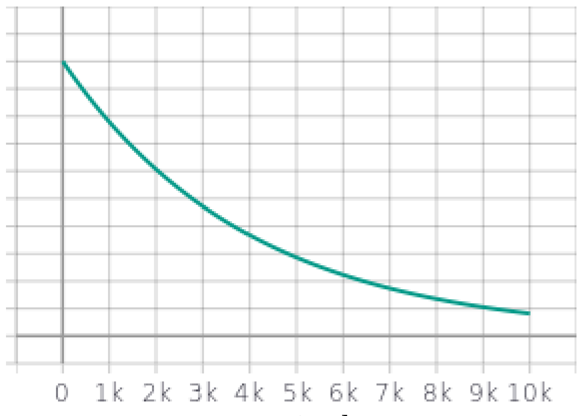
Episodes

Average reward



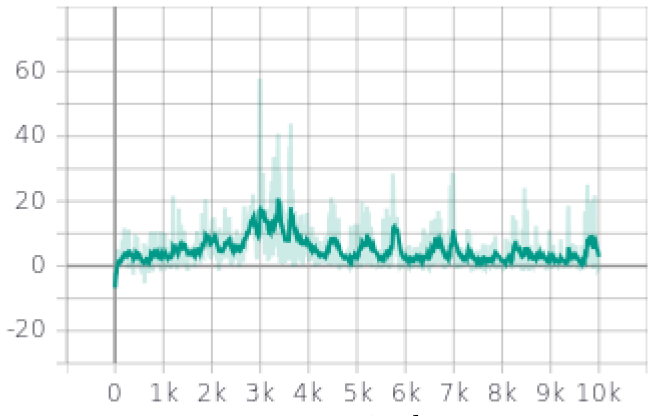
Episodes

Epsilon decay



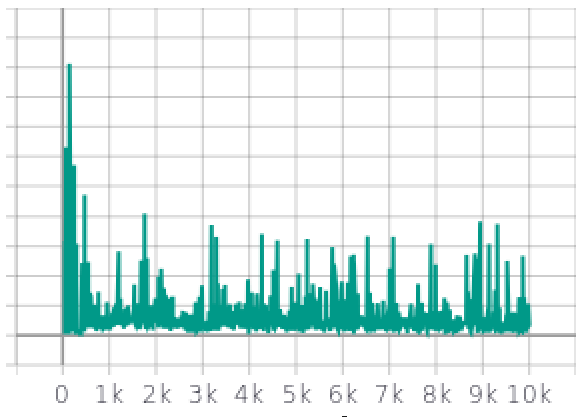
Episodes

Maximum reward



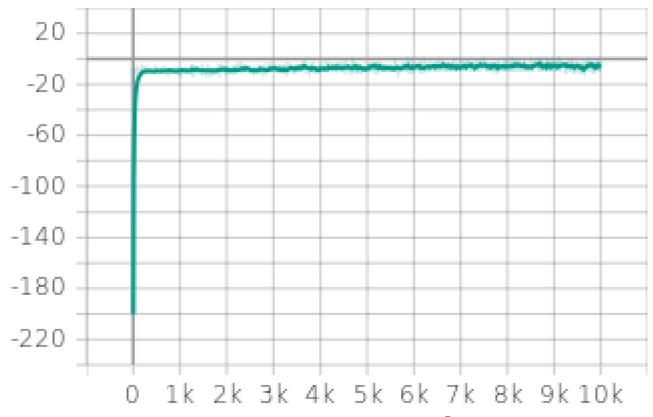
Episodes

Loss



Episodes

Minimum reward



Episodes