Department of Instrumentation and Control Engineering
National Institute of Technology Tircuhirappalli

B.Tech Final Year (2019-23 Batch) Project Work (ICIR17) - Second Review

Project Title: Reinforcement learning for self driving cars in CARLA simulaton.

Students Roll no & Name: 110119012 - Rakesh Akkannagari.

110119043 - Sri Sreya Grandhi.

110119048 - K.Deekshitha Hima Priya.

110119054 - Kokilavayamatham Spoorthi.

110119120 - Thummala Greeshma.

Guide Evaluation:

1. No of times meeting (Online and offline) happened with the students after first review:

   Remarks:

2. Roll number of Absentees for the meetings:

   Remarks:

3. Is the project work progressing as per the approved plan Yes/ No

   Remarks:

4. Rate the student based on the contribution to the project work in a scale of 10 to 1, 10 being good & 1 being very poor.

	Roll No	Mark (10)
1		
2		
3		
4		
5		

Name and Signature of the Guide

Department of Instrumentation and Control Engineering
National Institute of Technology Tircuhirappalli

Second Review Report

Title: Reinforcement learning for self driving cars in CARLA simulaton.

Guide Name: S.Narayanan

Team Members

Roll No	Name	Signature
110119012	Akkannagari Rakesh	
110119043	Sri Sreya Grandhi	
110119048	K.Deekshitha Hima Priya	
110119054	Kokilavayamatham Spoorthi	
110119120	Thummala Greeshma	

Objectives and Outcomes

The main objectives of a project may include:

- Developing an AI system that can effectively navigate complex and dynamic environments: The primary objective of a self-driving car project is to create an AI system that can make decisions in real-time and navigate safely and efficiently through different environments.
- The goal 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.
- Developing a prototype in a simulator: To deploy self-driving cars on a large scale, the AI system must be scalable and robust. Reinforcement learning can help the system to adapt to new environments and handle unexpected situations, such as adverse weather conditions or roadblocks.
- Overall, the objective of a project focused on developing self-driving cars using reinforcement learning that can get adopted to the new environment conditions.

Work Completed

1. Setting up the Carla environment.
   Download and install CARLA: The first step is to download the CARLA simulator from the official website and follow the installation instructions for your operating system. (CARLA version 0.9.8)

   Set up Python API: CARLA provides a Python API for controlling the simulator and interacting with the environment. To set up the Python API, you need to install the CARLA Python package using pip. This package provides the necessary modules and classes for communicating with the simulator.

   Configure the environment: Before you can run your first simulation, you need to configure the CARLA environment by specifying the desired settings, such as the map to use, the number of vehicles and pedestrians, weather conditions, and other parameters. You can create a configuration file in Python or use one of the pre-defined configurations provided by CARLA.

   Start the simulator: Once you have set up the environment, you can start the CARLA simulator and connect to it using the Python API. This allows you to control the simulation, retrieve data from the environment, and interact with the simulated vehicles and pedestrians.

2. Training the model

We have planned for a series of experiments with different reward function for our DQN agent, adding some new parameter to improve the model performance.

- Firstly, with a basic reward function with three actions. (Move forward, left and right turns) Negative rewards for collision and low episode speed, positive rewards for good episode speed, defined a threshold to speed and assigned accordingly.

We went with the Xception model, since there are articles where it has outperformed normal CNN's.

For rewards, we setup the following:

- +1 for each frame driving > 50KMPH
- -1 for each frame driving < 50KMPH
- -200 for a collision and episode is over

The first thing we discovered was that both loss and q values were basically exploding.

Check the results down in Results section.1

This was obviously, due to the monumental size of the crash penalty in comparison to all else. Also possibly the bounds being out of range. For example, we might have had more success by doing something more like:

- +0.005 (1/200) for each frame driving > 50KMPH
- -0.005 (-1/200) for each frame driving < 50KMPH
- -1 (-200/200) for a collision and episode is over

But instead we just went with:

- +1 for each frame driving > 50KMH
- -1 for each frame driving < 50KMH
- -1 for a collision and episode is over

This seemed to curtail the explosiveness of Q values and loss, but we still found agents would undoubtedly go with only 1 action constantly when given 0 epsilon.

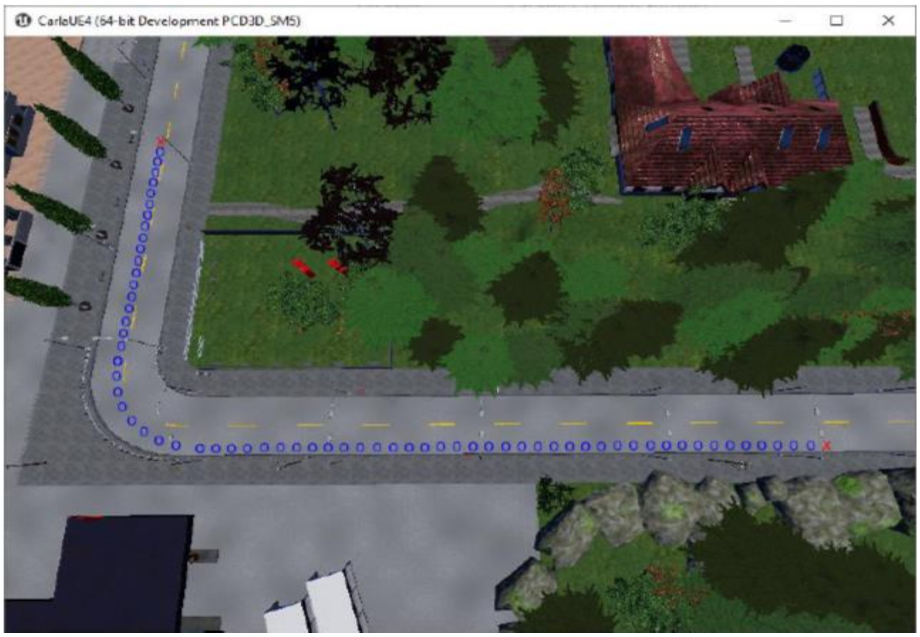
Reinforcement learning is quite a bit different from supervised learning, mainly in the fact that supervised learning is pure ground-truth (or at least that's the expectation). All imagery you feed it, and the labels, are meant to be 100% accurate. With reinforcement learning, this isn't really the case. We're fitting a model, yes, but we're also fitting for these Q values. It's far more complex operation going on, and things are going to be a bit “fuzzier” to the model. No reason to also have a highly complex neural network.

We then tried with a simple 64*3 CNN, it performed way better than Xception. Previously the accuracy of the agent is quite high which simply means that it is overfitting every time and now it is relatively better also there are no explosions in loss.

Check the results down in Results section.2

This is just making the car to move around the city but not actually it is moving from source to destination. For this,

- Second, is to train the agent between some source and destination by adding some waypoints in between.



This is the sample layout of the waypoints each with 2meters distance, the agent has to follow the end points with each waypoint the agent getting closer it gets a positive reward.

Rewards

Reward 1: Collision: -20 points
Reward 2: Arrived at destination: +20 points
Reward 3: If distance to destination decreases: +5
Reward 4: Short training episode (<= 20 seconds): -10
Reward 5: Others: -1

Check the results down in Results section.3

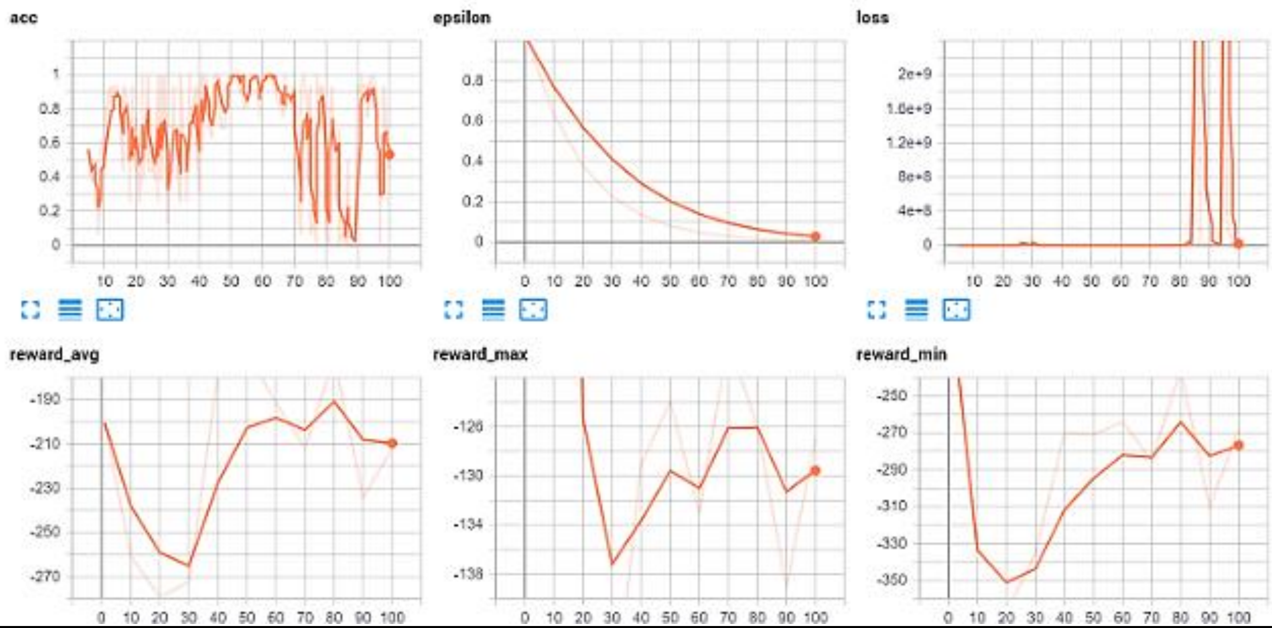
In this training we can see a great improvement in the episodes. Now we can see a big upward trend in the graph of the average reward and in the maximum reward. This indicates that the car itself is learning to get better rewards. At this point, the car has a very simple waypoint system, at each step 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.

Yet to be completed

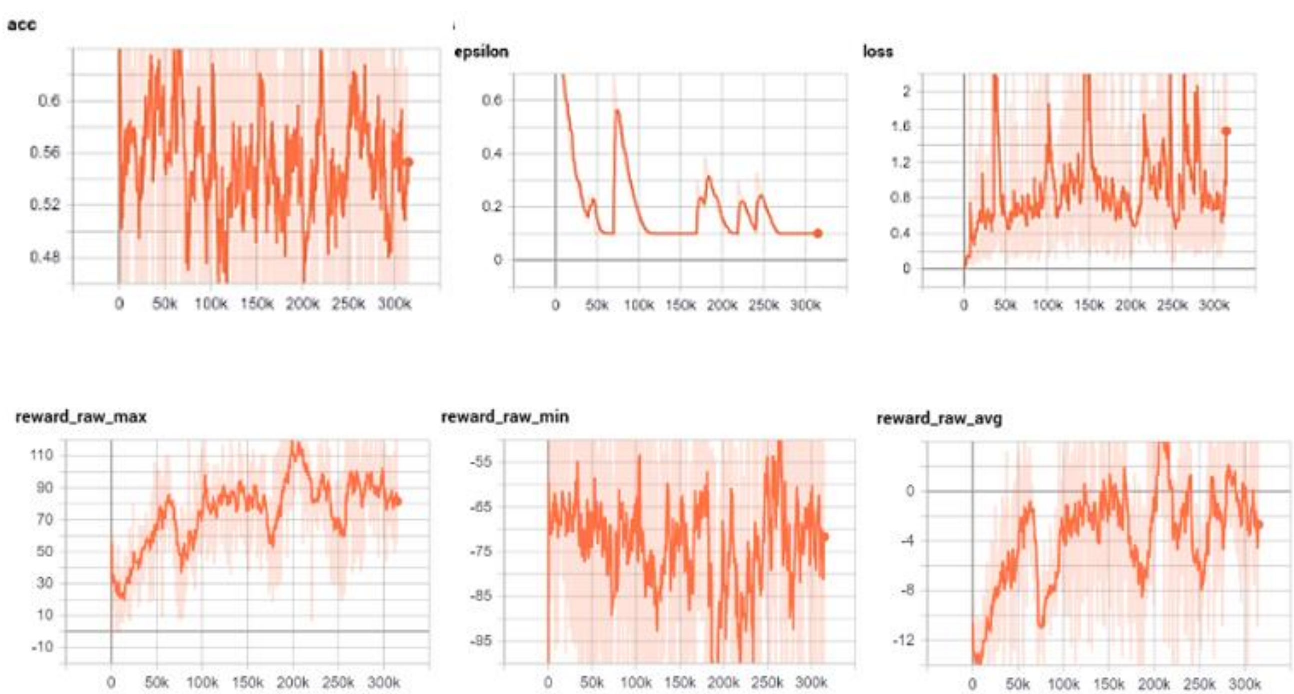
- Train agent to track lanes.
- Train with other cars and pedestrians.

Results

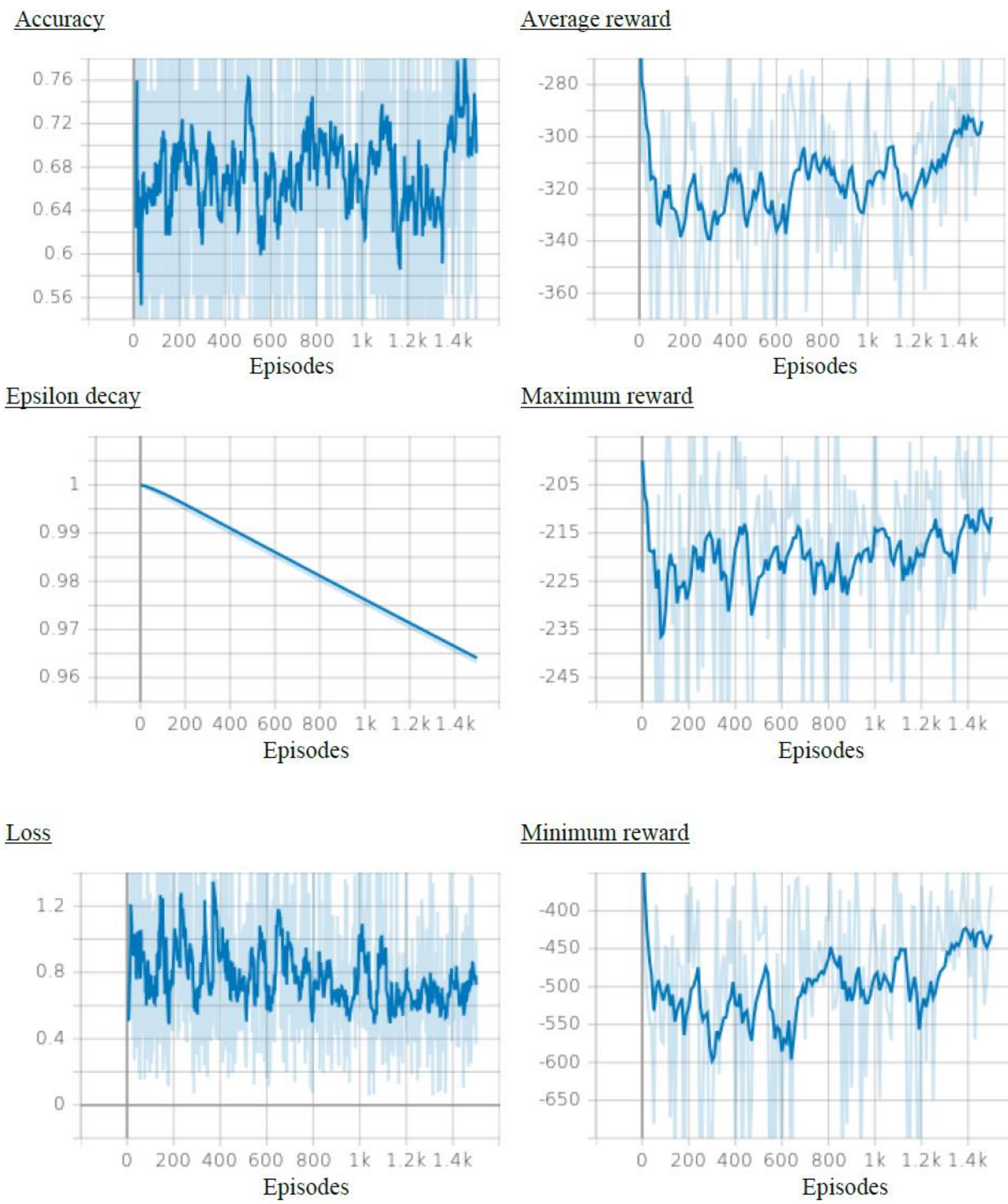
1.


acc
epsilon
loss
reward_avg
reward_max
reward_min

2.


acc
epsilon
loss
reward_raw_max
reward_raw_min
reward_raw_avg

3.



Signature of the Guide