ENPM 690: Final Project Progress Report

Topic: Highway Intersection Turning in Self-Driving Car Using Deep Reinforcement Learning

Prepared By:

Pradip Kathiriya (117678345)
Rutvik Kevadiya (118158440)

Contents

1. Introduction

A widespread approach of AI application for self-driving cars is the Supervised Learning approach and, above all, for solving perception requirements. But a self-driving car is very similar to a robot and an agent in a Reinforcement Learning (RL) approach. Can we replace a supervised learning approach with a reinforcement learning approach? The disadvantage of a supervised approach is a human bias involved in the entire AI process, from data collection to model deployment.

Interacting with the environment is the most significant task of a self-driving car. Perception is the first step, which is currently AI-based, and a supervised approach is applied. In this approach, you need to consider that the vehicle is driving in an open context environment, and you need to train your model with all possible scenes and scenarios in the real world. The variety of scenes and scenarios is the main difficulty that Tesla, Waymo, and Cruise must solve by collecting more and more data and validating system operations based on the collected data. How can we ensure that a self-driving car has already learned all possible scenarios and safely masters every situation?

Reinforcement Learning (RL) could be a solution to this problem. An RL approach means that an agent gathers the environmental information and switches from one state to the next state, based on a defined policy to maximize the rewards. What actions does an agent take on as the brain of a self-driving car? To keep it simple, three actions of acceleration, deceleration, and steering are the most critical actions that affect vehicle dynamics and road safety. The riskiest decision is steering, and the least critical one is braking.

In this project, we have developed the reinforcement learning pipeline to teach the riskiest decision i.e., steering to the autonomous car. We have emulated the highway intersection crossing scenario of a self-driving car and developed a pipeline to teach the car how to take the left turn without colliding with other vehicles. The goal here is to take a left turn and continue driving without any collision. We have defined a policy that rewards the agent (i.e., car) if it successfully takes the left turn without any collision and penalizes the agent if it collides during the turning. We also reward the agent if it accelerates and complete the turn in less time.

2. Environment Set-Up

2.1 Prerequisite

This project requires python3 (>=3.5)

This project required the Installation of pygame:

```
python3 -m pip install -U pygame
```

This project requires the installation of OpenAI Gym:

```
pip install gym
```

2.2 Installation

```
pip install highway-env
```

2.3 Snapshot of Environment

Below is the snapshot of the environment that we are using for this project

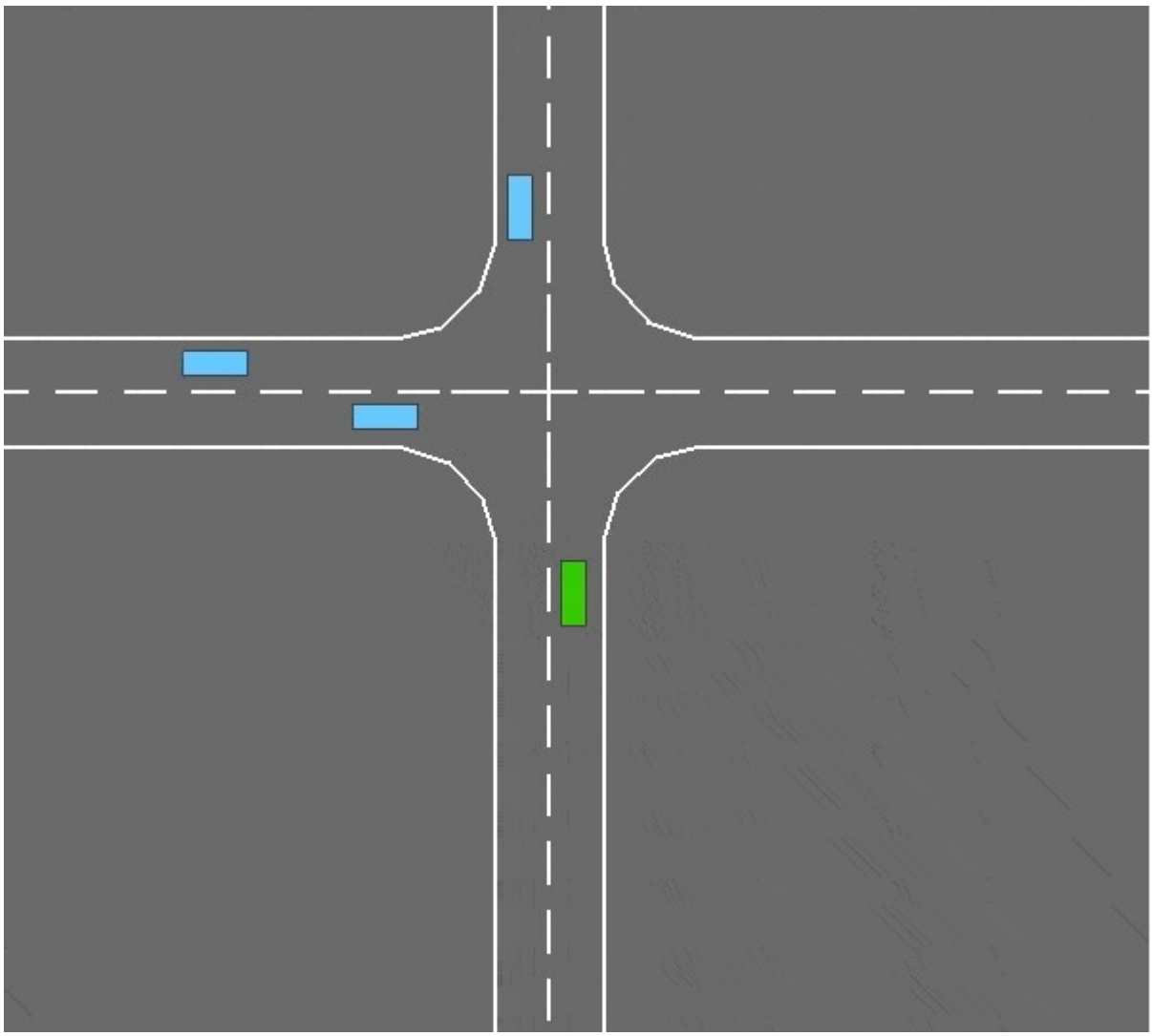


Fig : Snapshot of the environment

3. Current status

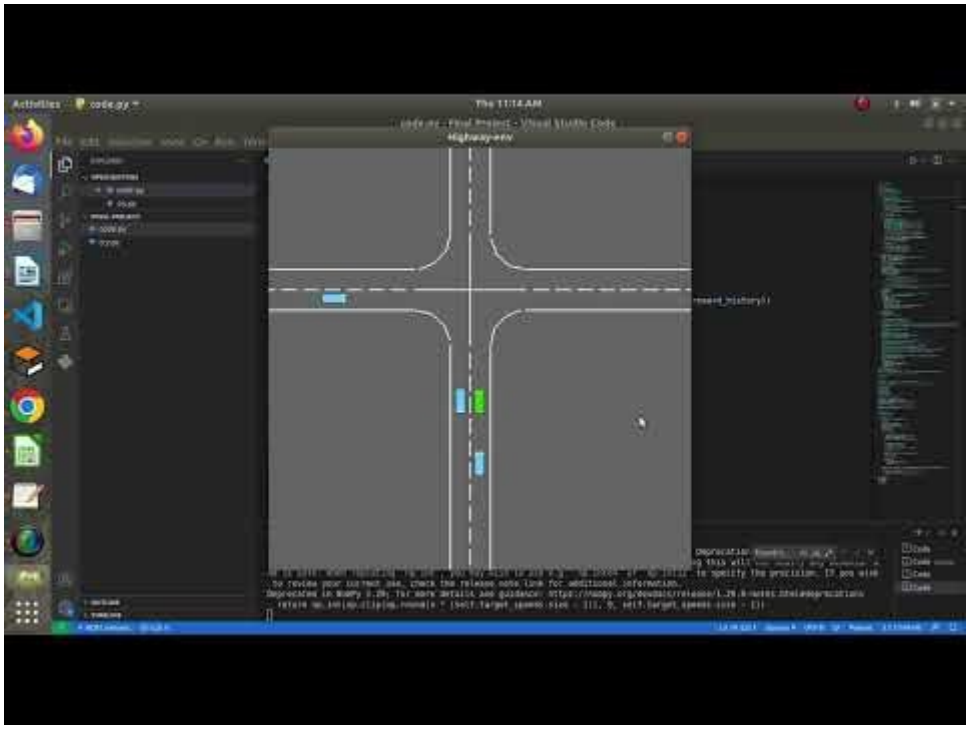
3.1 What have we done so far?

1. Set up environment – Completed
2. Define a network (Policy and target network) – Completed
3. Define reward function and learning scheme – Completed

3.2 What is remaining?

1. Hyper parameter tuning of the network
2. Correcting the behaviour of other agents (currently, other vehicles are colliding with each other)
3. Debugging

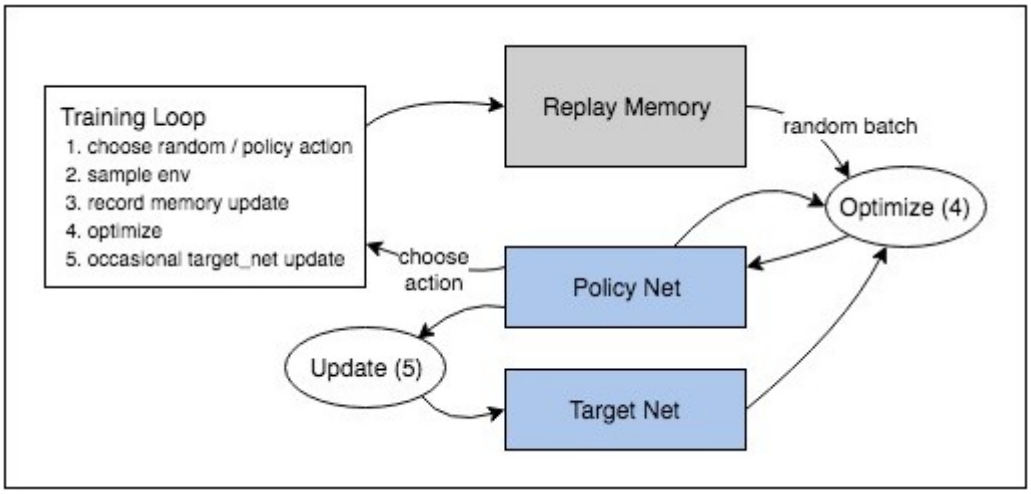
3.3 Video of current Progress



Video: Current of current Learning scheme
(Link: https://youtu.be/HDYkHxg7YJY)

4. Methodology

We are using DQN (Deep Queue Network) for this project. Below diagram is illustrating overall learning process of the model.



Learning scheme consists of two classes: one for reply memory and other for the policy network and target network and loop that encapsulate learning process.

Reply memory are used to store all the experiences that the agent has with the environment. Reply memory are equivalent to data set in the supervised learning. Data set is randomly selected from reply memory to feed it into the neural network for learning process. Data set is selected randomly to avoid correlation between the data point.

Policy network and Target network are essentially the same network with the only difference is the update technique. We are using Epsilon-Greedy Policy during the learning process to avoid sub-optimal solution.

In the learning loop, at each step, first check whether to explore or exploit using Epsilon-Greedy policy. Execute the selected action to obtain the next state and reward. In this project, if the vehicle successfully crosses the intersection, then the reward is +1, if the vehicle crash with another vehicle, the reward is -5 and if the vehicle accelerates then the reward is +1. Store this information in reply memory. Take a mini batch as a random sample from the memory and compute the target Q value using the target network and compute the predicted Q value using the policy network. Take the difference between the target Q value and predicted Q value and back-propagate the loss into the policy network to update the weight of the networks. Also at regular intervals, copy the weight of the policy network into the target network.

5. References

1. https://github.com/eleurent/highway-env
2. https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html
3. ENPM 690: Robot Learning – Class notes