Project Title : EndGame

Car Navigation using TD3

(Twin Delayed Deep Deterministic Policy Gradient) Deep Reinforcement Learning Algorithm:

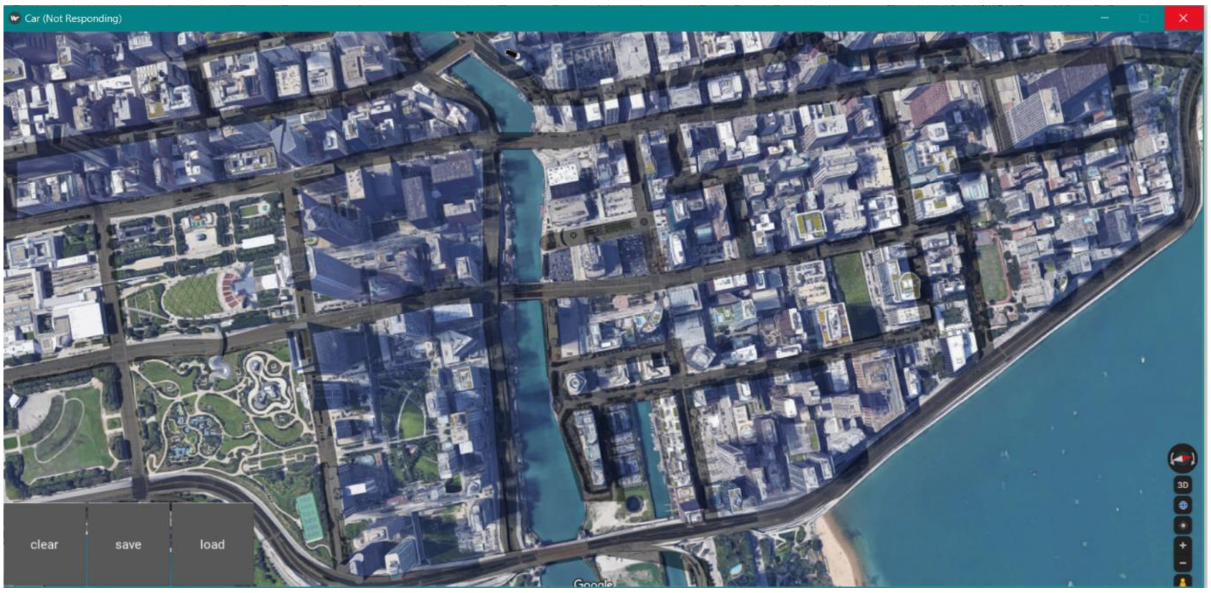
Submitted by : Chaitali Deb Purkayastha

Email id : chaitalidebp@gmail.com

Extensive Vision AI (EVA) TSAI Final Project

I. Overview

Problem Statement



A city map to traverse and a small toy car replica was provided.

Objective:

1. Car should be properly driving through the road .
2. Car should be able to reach the goal with minimum steps.

As guided , Kivy Environment is used:

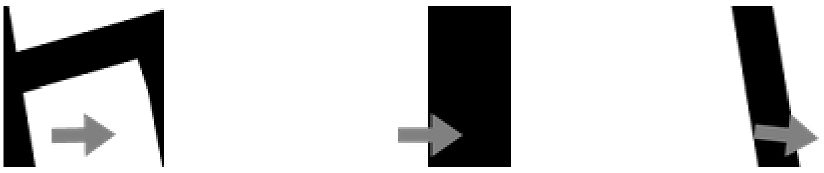
https://kivy.org/doc/stable/installation/installation-windows.html

II. Environment

a. State Space

State space consists of

i. A 80x80 numpy array from the sand of the city. An arrow is superimposed giving the direction of car going. The following images show samples of how the state looks like



ii. The following additional attributes are also added to state space

Attribute	Description
Angle	Angle of car
Alignment	Alignmentof car to the goal
On Road	Whether the car is on road. 1 If on road, 0 if off road
hit boundary	Has the car hit boundary,. I if yes, 0 for no
Hit goal	Has the car hit goal. 1 if yes, 0 for no
Distance diff	Normalized Difference of distance of car position to the goal and the last position to the goal
Distance Reduced	Is the distance reduced. 1 for yes, 0 for no

b. Action Space

Attribute	Description
Rotation	How much to rotate the car. Continuous value between -3 to 3
Velocity	Car displacement. Continuous value between 0.4 to 2.4

a. Reward Structure

Condition	Reward	Comment
Car is off the road	-3	
Car is on the Road but distance to Goal is reduced	7.0	
Car is on the Road but distance to Goal is increased	4.5	
Car Hit the Boundary	-50	Car moves randomly, if hits boundary.
Cat Reaches the Goal	+100	Once car reaches one goal, another goal is assigned.
Living Penalty	-0.1	

Each Episode has fixed number of 2500 steps. Once episode is completed, done variable is set to True.

II. Proposed Algorithm to achieve the desired Objective

After considering all aspects of the problem statement,
Twin Delayed Deep Deterministic (TD3) algorithm (https://arxiv.org/pdf/1706.02275.pdf) is used to achieve the desired objective. TD3 is an off policy algorithm which can be applied for continuous action spaces.

TD3 uses Actor and Critic principle. TD3 uses two Critic Networks

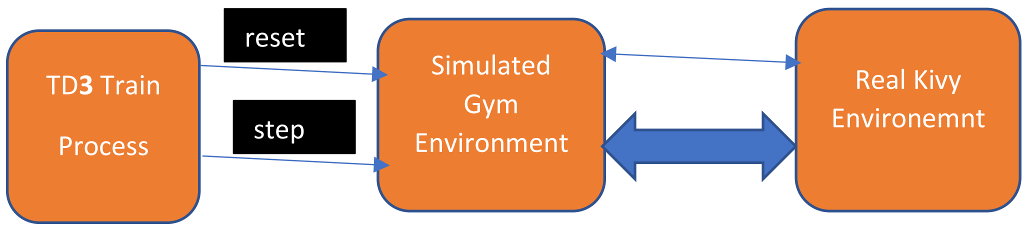
TD3 uses experience replay where experience tuples (S,A,R,S`) are added to replay buffer and are randomly sampled from the replay buffer so that samples are not correlated.

TD3 algorithm also uses separate target neural network for both Actor and Critic for each of the agent. As target values are determined for both the critic and actor networks, copy of both of these networks and soft update their weights are periodically updated to the respective target networks using polyak averaging

III. Methodology and Solution approach

i. Simulated Gym Environment to encapsulate the Kivy Environment

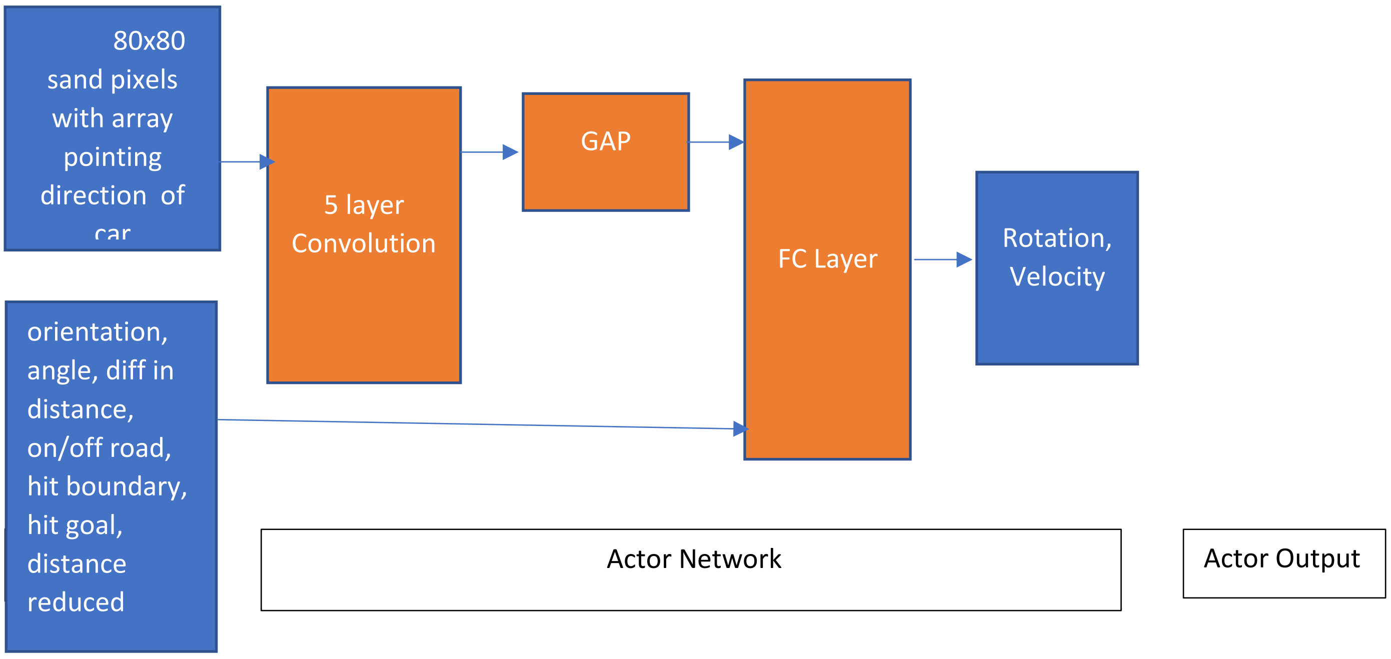
Kivy Environment does not provide methods like reset, step which is very easier to work for any RL project. To solve this I created a simulated Gym Environment which interacts with Kivy based on Multiprocess Queue and Event mechanism provided by Python. The real Kivy environment works on a separate process while TD3 training works on a separate process.



In this TD3 train process first starts and it will start the Kivy Environment. There is simulated gym Environment to which TD3 Train process can call methods like env.reset() to reset the environment and env.step(action) to take a action ands gets next state.

Internally Simulated Gym Environment interacts with Real Kivy Environment using Event and Message Queues.

ii. Actor Network


80x80 sand pixels with array pointing direction of car
orientation, angle, diff in distance, on/off road, hit boundary, hit goal, distance reduced
5 layer Convolution
GAP
FC Layer
Rotation, Velocity
Actor Network
Actor Output

Actor Input:
The Actor Network takes Input as two element tuple

i. first element is a 80x80 Numpy array representing the pixel values of sand 40 pixels around the car position with a arrow superimposed showing the direction of
ii. Second element is a Numpy Array having 4 parameters, these are
   a. Orientation of car to the goal
   b. Angle of car
   c. Difference in distance between current car position to the goal and previous car position and the goal divided by 4
   d. A flag on_road, whose value 1 means car is on road and 0 means car is off the road
   e. A flag hit boundary
   f. A flag hit goal
   g. A flag distance reduced

Convolution Layer:

There are 5 convolution layers used to transform the road pixel input. Except last layer, each layer 24 3x3 filters with stride 2 and ReLU Activation is used. Last layer has 24 3x3 filter with stride 1

GAP Layer:

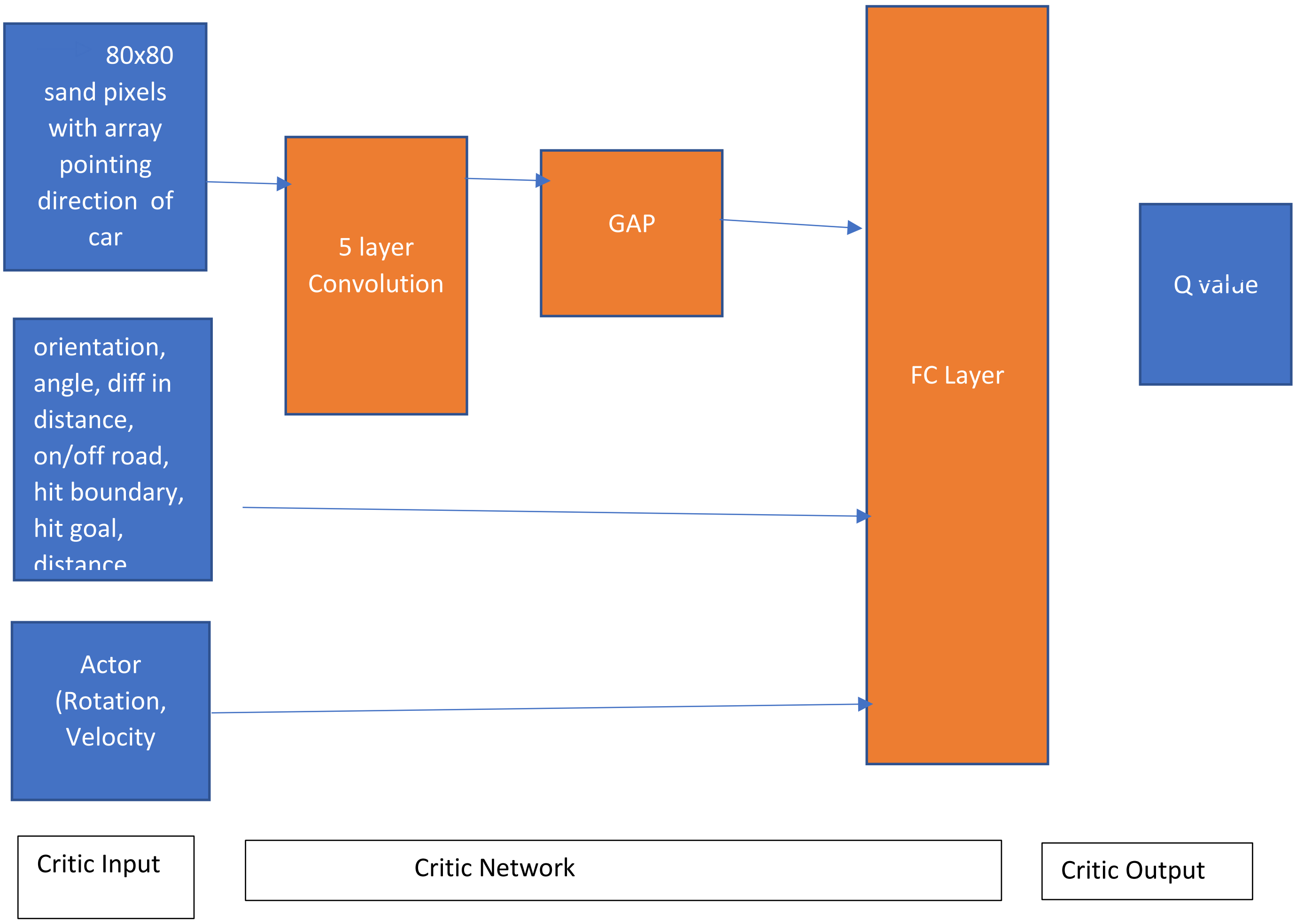
Global average pooling layer is added after 5 convolution layer which transform into 24x1x1

FC Layer:

There are three full connected layers.

- First layer layer takes the GAP output flattened and concatenate the 7 extra parameters and output is 80 1d tensor
- Second layer takes first FC layer input and output is 144 1d tensor and applied ReLU activation
- Third layer output form second layer transform to 1 tensor on which tanh is applied and multiplied by max_action to get the actor output

iii. Critic Network


80x80 sand pixels with array pointing direction of car
orientation, angle, diff in distance, on/off road, hit boundary, hit goal, distance
Actor (Rotation, Velocity
5 layer Convolution
GAP
FC Layer
Q value
Critic Input
Critic Network
Critic Output

Critic Input:
The Critic Network takes Input as two element tuple

iv. first element is a 80x80 Numpy array representing the pixel values of sand 40 pixels around the car position with a arrow superimposed showing the direction of

v. Second element is a Numpy Array having 4 parameters, these are
   h. Orientation of car to the goal
   i. Angle of car
   j. Difference in distance between current car position to the goal and previous car position and the goal divided by 4
   k. A flag on_road, whose value 1 means car is on road and 0 means car is off the road
   l. A flag hit boundary
   m. A flag hit goal
   n. A flag distance reduced

vi. Action output (Rotation, Velocity)

Convolution Layer:

There are 5 convolution layers used to transform the road pixel input. Except last layer, each layer 24 3x3 filters with stride 2 and ReLU Activation is used. Last layer has 24 3x3 filter with stride 1

GAP Layer:

Global average pooling layer is added after 5 convolution layer which transform into 24x1x1

FC Layer:

There are three full connected layers.

- First layer layer takes the GAP output flattened and concatenate the 7 extra parameters and action (rotation, velocity) and output is 80 1d tensor
- Second layer takes first FC layer input and output is 144 1d tensor and applied ReLU activation
- Third layer output form second layer transform to 1 tensor

which represents Q value

i. Hyper parameters Used

We have used Adam optimizer for both Actor and Critic Networks with the following hyper parameters

Batch Size: 128
Discount Rate (gamma) : 0.99
Soft update of target parameters (Tau) : 0.005
Initial warmup episodes without learning: 10000 timesteps
Number of learning steps for environment step : 3
Exploration Noise : 0.1
Policy Noise : 0.2

ii. Techniques used to Improve Learning

To improve the learning I have used the following techniques

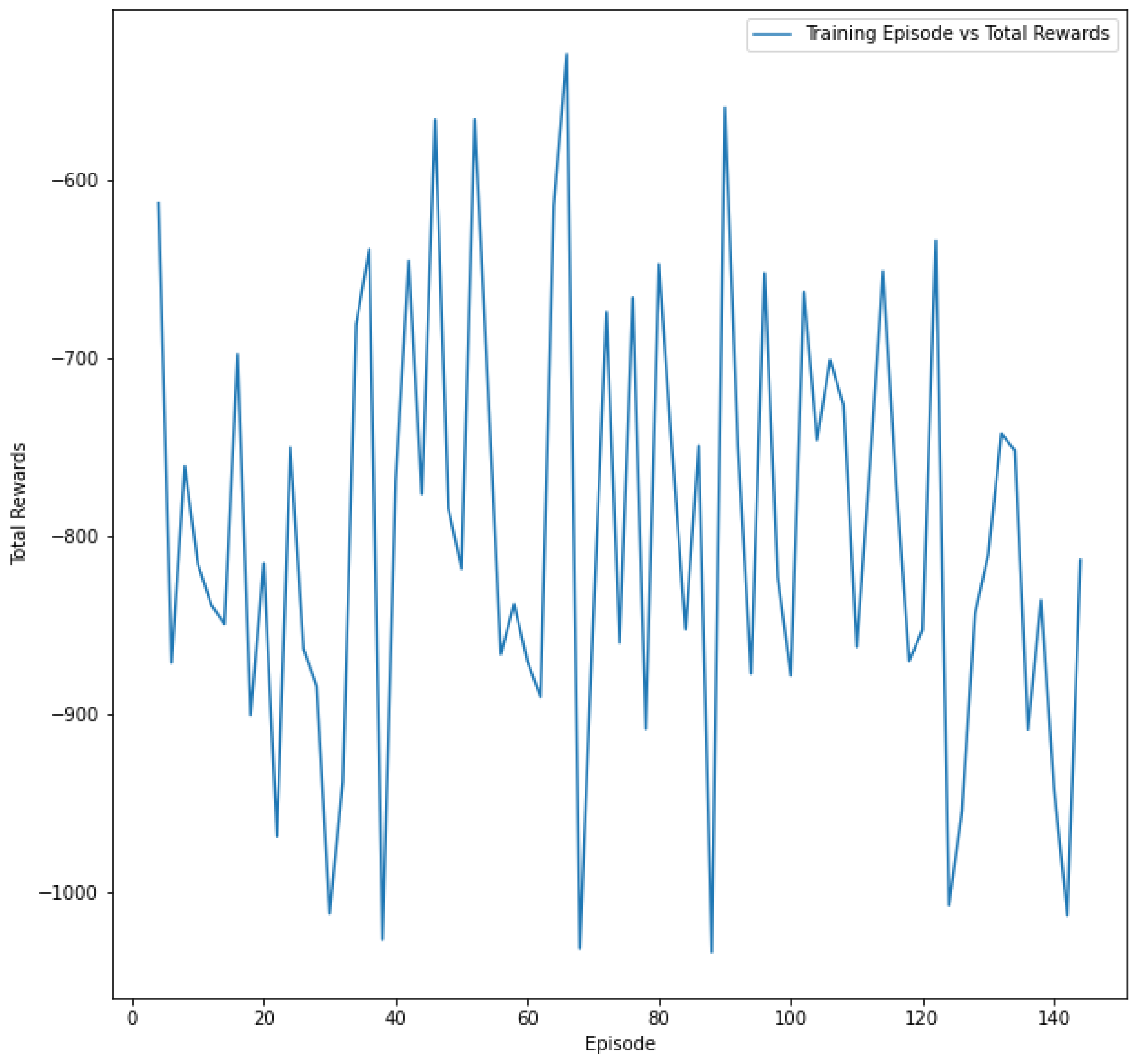
i. After every 2 training cycle, evaluations on policy is done for 500 steps for 10 times
ii. Initial 10000 timestamps the replay buffer is filled up random policy by choosing action randomly from the action space. This will help better exploration
iii. For episode explorations, I used a epsilon value which is initialized to 0.9 and over 40 episodes, I reduce the value to 0.2. A random number is generated between 0 and 1 if it is less than epsilon value then next action is taken from random policy else action is taken from the

iv. Gaussian noise with mean value of 0 and standard deviation (sigma) of 0.1 has been added to explore states.

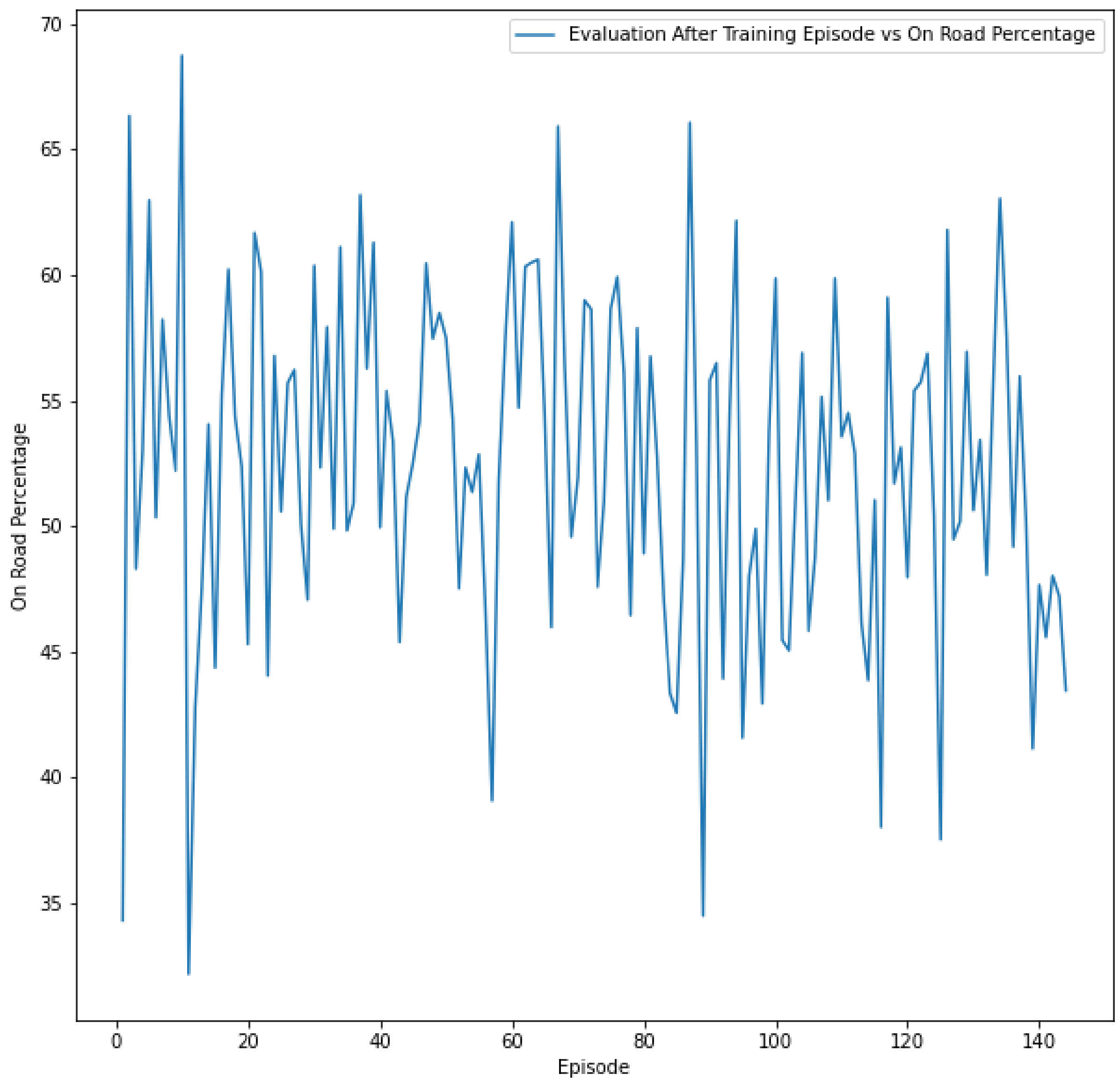
v. Ghoomar effect is overcome by using learning rate of Adam optimizer and l2 regularization

IV. Results

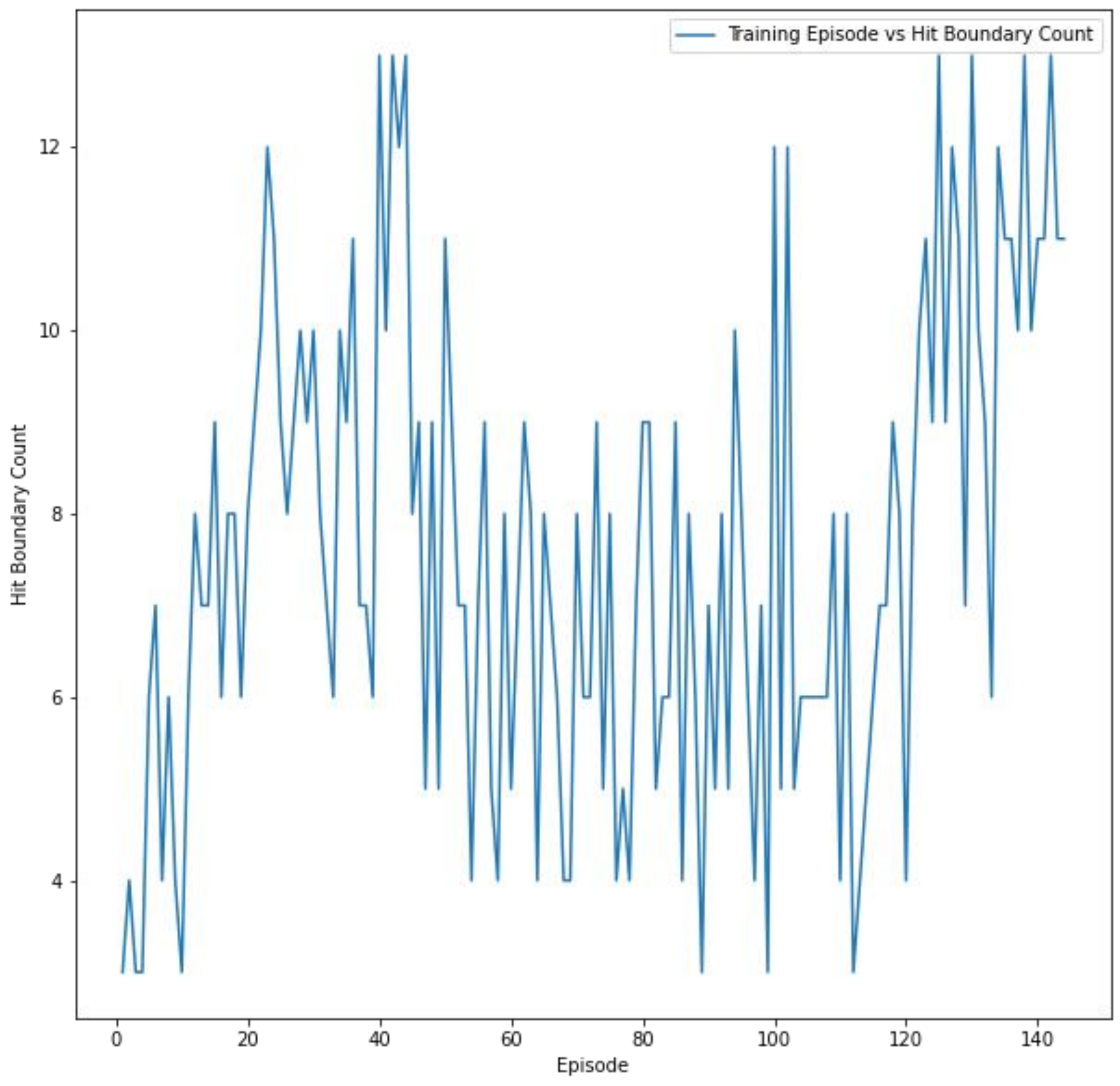
- **Plot for Training episode vs Total rewards**



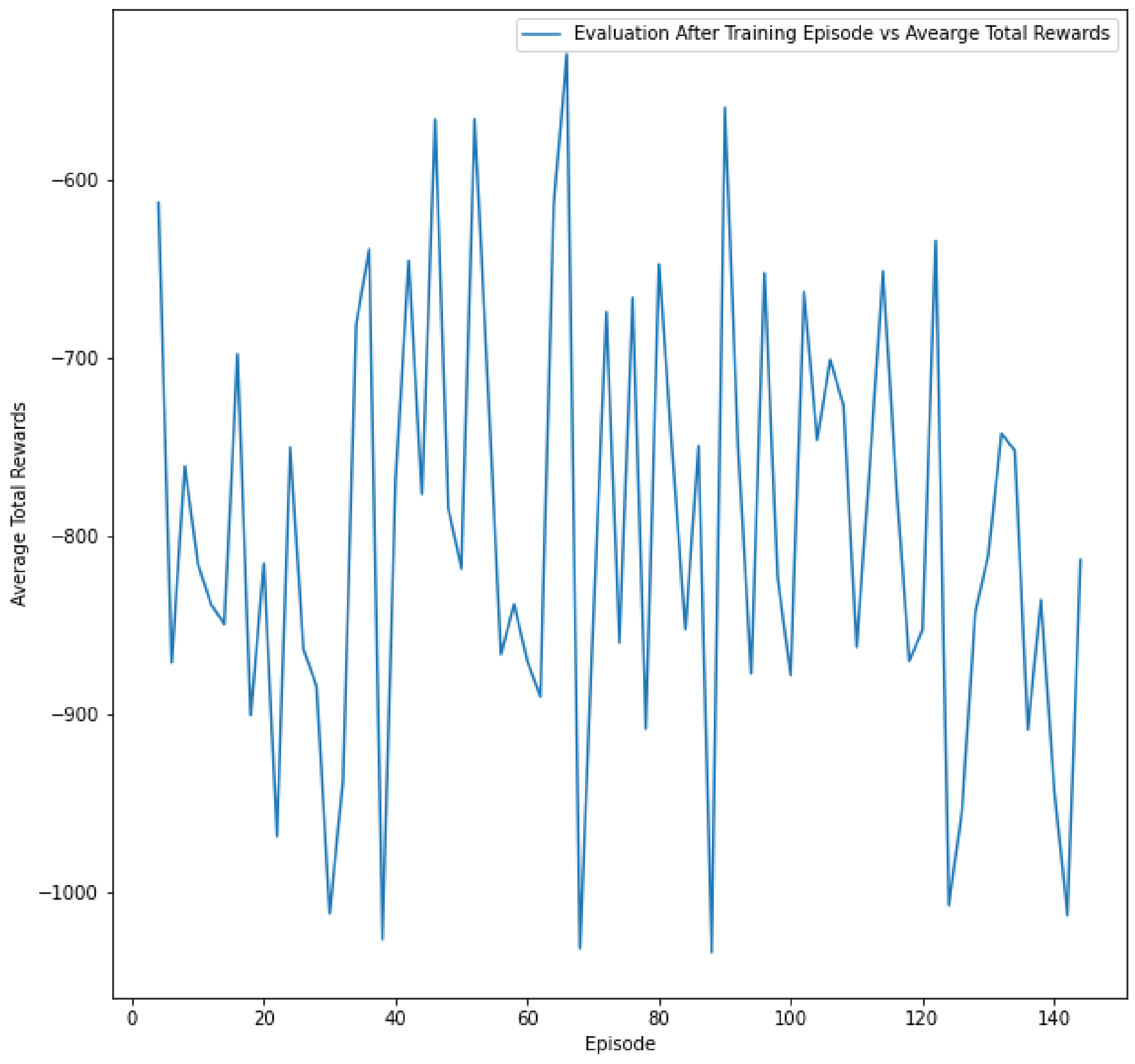
- **Plot for Training episode vs On road percentage**



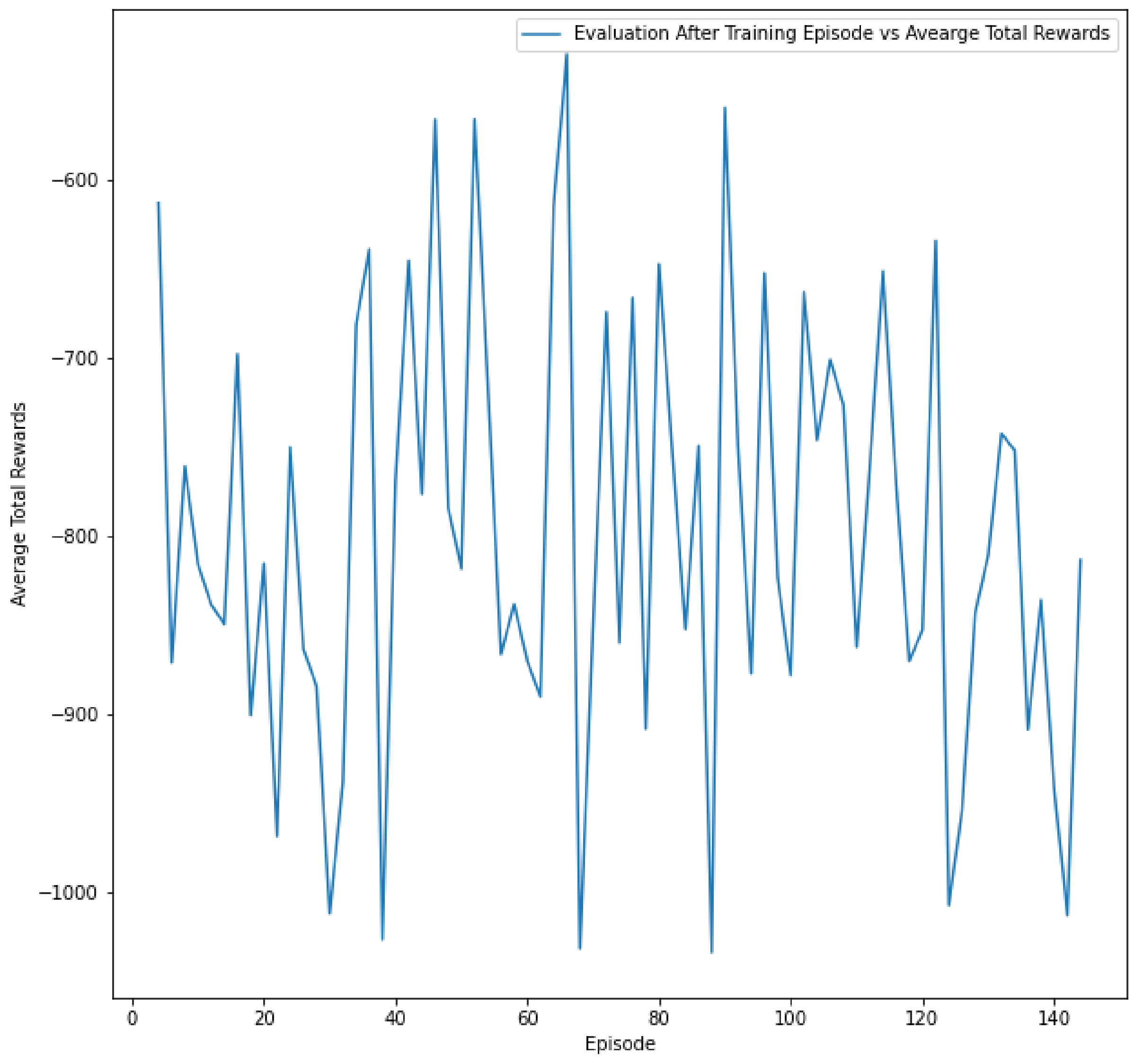
- **Plot for Training episode vs Boundary Hit Count**



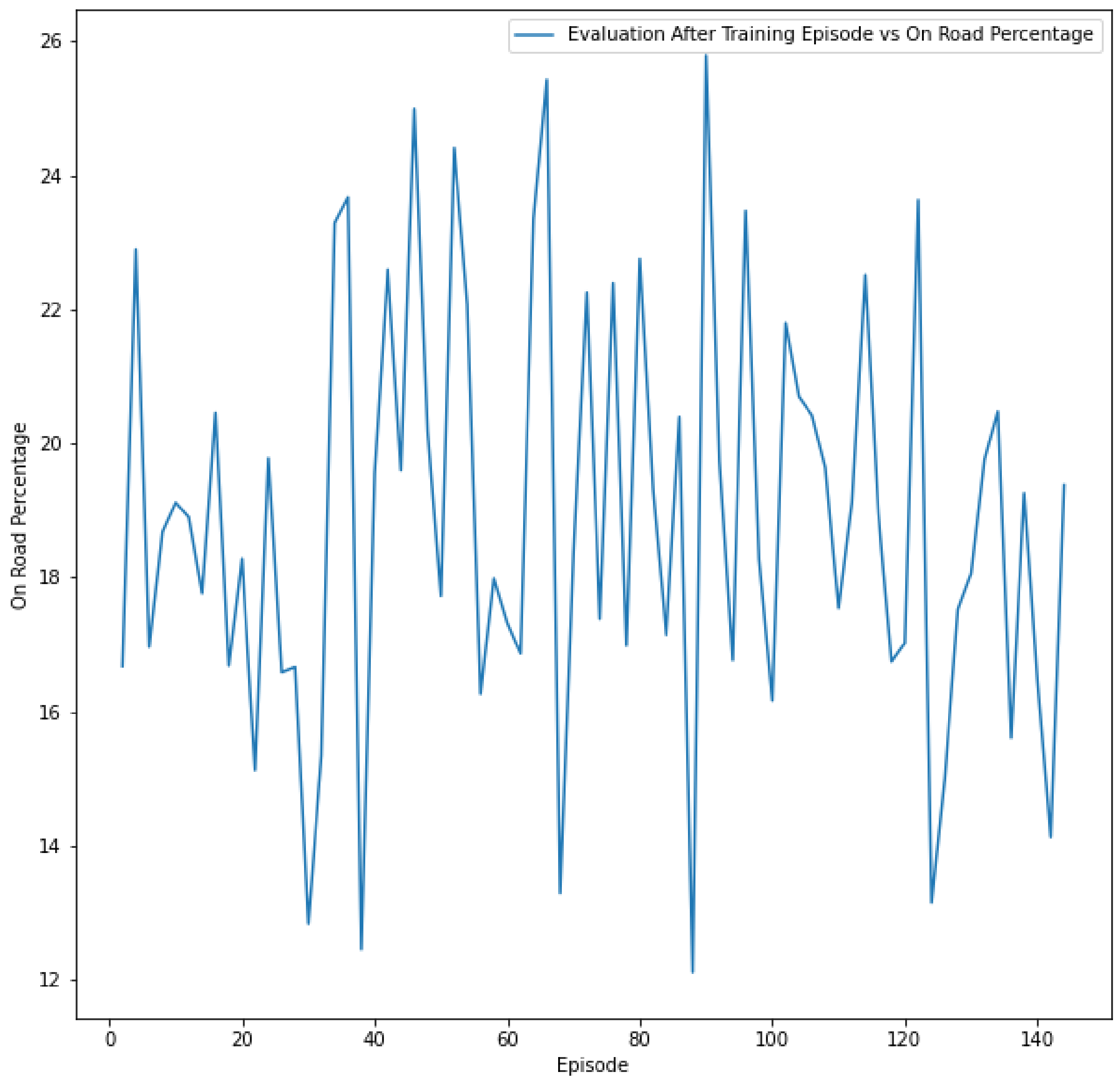
- **Plot for Evaluation Training episode vs Average Total rewards**



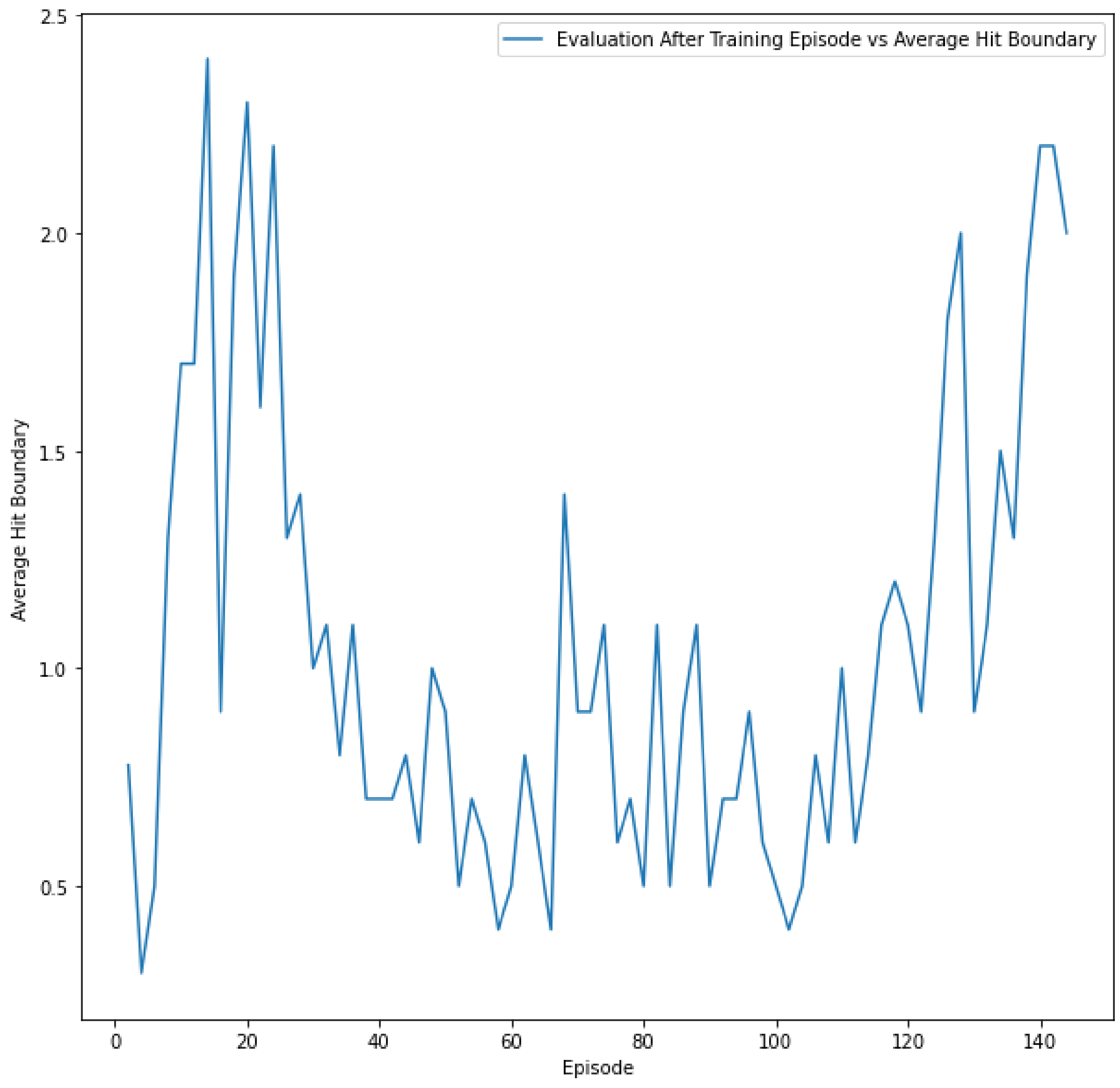
- **Plot for Evaluation Training episode vs Average Total rewards**



- **Plot for Evaluation Training episode vs Average On Road Percentage**



- **Plot for Evaluation Training episode vs Average Boundary Hit**



V. Code Structure

map.py - Python file for Kivy Environment for Car

TD3_Train.py- This is the main for training using TD3

TD3_Test.py- This is the main for training using TD3

SimulatedGymEnvironmentFromKivyCar.py – Simulated Gym environment from Kivy

generate_plot.ipynb – Jupyter Notebook for generating plots

V. Conclusion:

In this project we have used MADDPG algorithm with neural networks having hidden layer dimension of 128x256x128 for both Actor and Critic and for both the agents

i. We have trained the agent and achieved the desired mean score over 100 consecutive episodes of 0.5131 in 1174 episodes
ii. Scores (Rewards) earned is a left skewed histogram
iii. Mean score over consecutive 100 episodes are initially flat but increases exponentially after 600 episodes

iv. To improve learning we have used various techniques like Learning rate reduction if the mean score over 100 episodes does not change by 0.05 for 100 episodes, also added initial warmup period of 400 episodes when no learning happens

Further Improvements

Further improvements to the project can be done using:

i. Prioritized Experience Replay
ii. Robust Multi-Agent Reinforcement Learning via Minimax Deep Deterministic Policy Gradient (https://people.eecs.berkeley.edu/~russell/papers/aaai19-marl.pdf)