Deep Reinforcement Learning for Decision Making in Autonomous Driving

Amey Anil Deshpande*, Bhushan Ashok Rane†, Denny Boby‡, Yash Rajendra Patil§
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester, Massachusetts, United States
*aadeshpande2@wpi.edu, †barane@wpi.edu, ‡dboby@wpi.edu, §yrpatil@wpi.edu

***Abstract*—This project leaps into comparison of different architectures that combine Reinforcement Learning and Deep Learning methods namely Deep Q-Learning henceforth referred as DQN, Dueling Double Deep Q-Learning henceforth referred as DDDQN and Proximal Policy Optimization henceforth referred as PPO which were done in discrete Action Space. We also tried Deep Deterministic Policy Gradient henceforth referred as DDPG to explore Continuous Action Space. The problem statement included use of a highway-v0 that is a decision making environment for autonomous vehicles. During the study we understood that the performance of algorithm depends on the type of observation extracted from the environment. Training the DDPG agent was especially difficult as it first needed to learn to follow a lane. We compared the performance of the discrete action space algorithms and the best performing one was DDDQN. We faced some issues regarding to the reward assignment and environment configuration but once it we got a deeper understanding of it, we were able to iterate much faster.**

https://github.com/ranebhushan/cs525_project

I. Introduction

Amalgamation of Reinforcement learning and Deep Learning methods has become a powerful framework capable of learning complex decision policies in high dimensional environments.Reinforcement Learning deals with the problem of learning agent placed in an environment to achieve a goal. Autonomous Driving is a promising technology to enhance road safety to enhance road safety, ease the road congestion, decrease fuel consumption and free human drivers. In the architecture of the autonomous driving decision-making is a key component to realise the autonomy.Autonomous cars have to take correct and safe decisions for driving in an urban environment without colliding with other agents. Decision-making algorithms for autonomous driving consists of rule-based and imitation based algorithms. Learning based methods like Reinforcement learning, Machine Learning are also being used for decision making for autonomous driving. We explored Deep Reinforcement learning for decision-making for Autonomous cars and used OpenAI Gym based Highway Environment (1) for testing our algorithms. We selected following algorithms for implementation:

- Deep Q-Learning (DQN) in discrete action space
- Dueling Double Deep Q-Learning (DDDQN) in discrete action space
- Proximal Policy Optimization (PPO) in discrete action space
- Deep Deterministic Policy Gradient (DDPG) in continuous action space

We chose these algorithms so that we can explore the deep reinforcement learning methods for discrete as well as continuous action space, also value-based and policy-based algorithms.

II. Related Work

Existing reinforcement learning algorithms mainly compose of value-based and policy-based methods. Vanilla Q-learning was first proposed in 1989 and then became one of the popular value-based methods. Recently a lot of variants of Q-learning algorithms, such as DQN (2), Double DQN (3), and Dueling DQN, have been successfully applied to autonomous driving cars. Different from value-based methods, policy-based methods learn the policy directly. In other words, policy-based methods output actions given the current state. Some of the policy-based methods like Deep Deterministic Policy Gradient (DDPG)(4), Soft Actor-Critic (SAC)(5) and Twin Delayed Deep Deterministic Policy Gradient (TD3) are also used in autonomous driving cars. In terms of autonomous driving, action spaces are continuous and fine control is required. All these policy-gradient methods can naturally handle the continuous action spaces. However, adapting value-based methods, such as DQN to the continuous domain by discretizing continuous action spaces might cause a curse of dimensionality.

In (6) the author explores the Behavior Planning of Autonomous Cars using Deep Reinforcement learning. He explores various Deep Reinforcement Learning algorithms for decision-making in highway-env. He also uses the Attention mechanism with Deep Reinforcement learning for decision-making in a complex environment. The author got good results by applying attention-based Deep Reinforcement learning.

In (7) the author incorporates safety mechanisms while training the agent in a highway environment. The author incorporates constraint optimization to add safety so that when the agent drives in the highway environment it understands that it has to incorporate safety so that no one gets hurt!

Thus, there is a lot of research where people are designing Deep RL algorithms for decision-making for Autonomous cars. Moreover, to make it robust researchers are mixing Deep RL algorithms with State of Art Prediction/Transformer based algorithms.

III. Methodology

A. Deep Q-Network (DQN)

DQN algorithm works according to the learn from trials method, the algorithm must train very many times against a situation and the algorithm will learn how to act in this environment to optimize the score.

In a specific case, the input of the Neural Network is an image at one moment in the game environment, and the output is illustrated by a layer of neurons with as many neurons as possible actions in the game. The neural network will try to understand what is the best action to take, the one that the algorithm will consider as the best is the one with the neuron with the highest value.

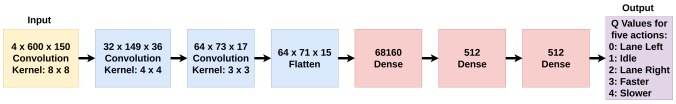

Fig. 1. Deep Q-Network (DQN) Block Diagram

To go into a little more detail in the total functioning of the program; first of all, it is necessary to build a database (memory buffer) composed of a maximum of 4 information sets: the current state (current image), the reward, the action and the next state (next image). Such kind of database is then used to train the neural network. Once the neural network is trained, just take any state (current image frame) of the environment, provide it as an input to the algorithm and then it will find the best action to take in this situation.

B. Dueling Double Deep Q-Network (DDDQN)

$$Q_1(S,A) \leftarrow Q_1(S,A) + \alpha(R + \gamma Q_2(S', argmax_a Q_1(S',a)) - Q_1(S,A))$$

Fig. 2. Double Q-Learning update

Dueling Double DQN involves some improvement techniques applied to the Deep Q-Learning method. Double DQN addresses the problem of overestimation in DQN. This overestimation occurs due to the presence of Max of Q value for the next state in the Q learning update equation. This max operator on Q value leads to maximization bias, which can result in bad performance by the agent in certain environments where the maximum of true value is zero, but the maximum estimate by the agent is positive.

Two different estimates as follows are present in Dueling DQN : 1. Estimate for the value of a given state: This estimates how good it is for an agent to be in that state. 2. Estimate for the advantage of each action in a state.

The above two estimates can be aggregated as :-

$$Q = V + A - \text{mean of all } A$$

C. Deep Deterministic Policy Gradient (DDPG)

DDPG (4) is a model-free off-policy actor-critic algorithm that combines Deep Q Learning(DQN) and a policy to iterate over actions. Original DQN works in a discrete action space

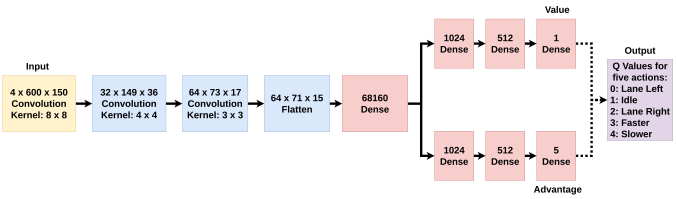

Fig. 3. Dueling Deep Q-Network (DDQN) Block Diagram

and DPG extends it to the continuous action space while learning a deterministic policy. As it is an off-policy algorithm, it uses two separate policies for the exploration and updates. It uses a stochastic behaviour policy for the exploration and deterministic policy for the target update.

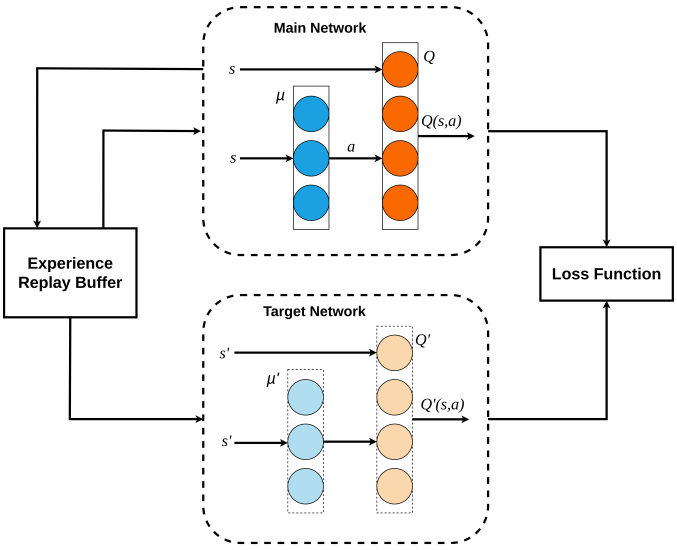

Fig. 4. Deep Deterministic Policy Gradient (DDPG) Block Diagram

DDPG is an actor-critic algorithm. Actor-Critic is a Temporal Difference(TD) version of Policy gradient. It has two networks: Actor and Critic. The actor decided which action should be taken and critic inform the actor how good was the action and how it should adjust. The learning of the actor is based on policy gradient approach. In comparison, critics evaluate the action produced by the actor by computing the value function. During the update process of the actor, TD error from a critic is used. The critic network gets updated based on the TD error similar to Q-learning update rule.

We did learn the fact that the instability issue that can raise in Q-learning with the deep neural network as the functional approximator. To solve this, experience replay and target networks are being used.

D. Proximal Policy Optimization (PPO)

PPO is a policy gradient method where policy is updated explicitly. We can write the objective function or loss function of vanilla policy gradient with advantage function.

The main challenge of vanilla policy gradient RL is the high gradient variance. The standard approach to reduce the

$$\nabla J(\theta) = E_{\pi_\theta}[\nabla_\theta log \pi_\theta(a_t \mid s_t)\hat{A}_t]$$

Fig. 5. Vanilla policy gradient expression with advantage function

variance in gradient estimate is to use advantage function. The advantage function estimates how good an action compared to average action for a specific state. The idea here is to reduce the high variance in the gradient estimate between the old policy and the new policy. The reduction of variance helps to increase the stability of the RL algorithm. We can write the advantage function as below:

$$A(s,a) = Q(s,a) - V(s)$$

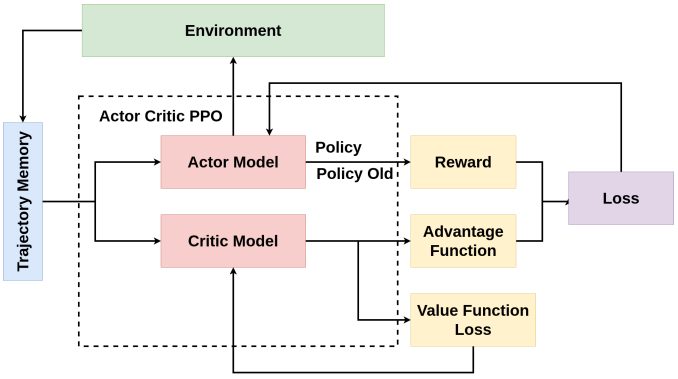

Fig. 6. Proximal Policy Optimization (PPO) Block Diagram

PPO is a first-order optimisation that simplifies its implementation. It defines the probability ratio between the new policy and old policy and we can call it as $r(\theta)$.

$$r(\theta) = \frac{\pi_\theta(a \mid s)}{\pi_{\theta old(a|s)}} \tag{1}$$

Without adding constraints, this objective function can lead to instability or slow convergence rate due to large and small step size update respectively. Instead of adding complicated KL constraint, PPO imposes policy ratio, $r(\theta)$ to stay within a small interval around 1. That is the interval between $1-\epsilon$ and $1+\epsilon$. ϵ is a hyper parameter.

$$J^{CLIP}(\theta) = E[min(r(\theta)\hat{A}_{\theta_{old}}(s,a), clip(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{\theta_{old}}(s,a))]$$

Fig. 7. PPO Objective Function

The objective function of PPO takes the minimum value between the original value and the clipped value. Positive advantage function indicates the action taken by the agent is good. On the other hand, a negative advantage indicated bad action. For PPO, in both cases, the clipped operation makes sure it won't deviate largely by clipping the update in the range.

IV. Environment

highway-env (1), a third party environment which is a part of OpenAI Gym, is used for implementation of autonomous driving decision making problem. This environment consists of multiple scenarios such as each has unique situations and continuous or discrete action space. The *highway-v0* environment that we used is a straight highway scenario in which the prime objective is to maintain speed and keep changing lanes for passing other vehicles.

- **Observation**: The environment exposes 4 observation types:
  - *Kinematics*: This observation gives the features like position, velocity and heading of the all teh vehicles present in the environment. the type of features can be configured using *features* parameter.
  - *Gray Scale*: This observation gives a gray scale image of the environment, the RGB to gray scale conversion uses *weights* parameter.
  - *Occupancy Grid*: This type of observation creates an occupancy grid around our agent and provides the features for the vehicles present in the grid.
  - *Time To Collision*: This observation provides an array that represents the predicted time-to-collision of observed vehicles on the same road as the agent.

  Each of the observation has it's own pros as well as cons. After exploring and experimenting we converged on fact that value based algorithms work better with *Gray Scale* observation type and policy based work better with *Kinematics* observation type. Notedly, Gray Scale observation needs a Convolutional Neural Networks for the agent to make sense of the observation while the case with other observation types is different, a multi-layered neural network can be used for all other observations. Once the observation is obtained it is passed to agent to take appropriate action, the environment allows discrete as well as continuous action space.
- **Actions:** When *DiscreteMetaAction* is used the actions available are:
  - 0 : *"Change the lane to one to the left"*
  - 1 : *"Stay idle"*
  - 2 : *"Change the lane to one to the right"*
  - 3 : *"Go Faster"*
  - 4 : *"Go slower"*

  For *ContinuousAction*, the environment has a continuous action space which consists of acceleration α and steering angle δ both ranging from -1 and 1 inclusive. By controlling both the values the agent can change lanes and accelerate or decelerate.
- **Reward**: The goal is to make agent focus on two tasks, the agent should move as fast as possible and progress quickly, and avoid collisions. Hence, reward function contains both the factors:

$$R(s,a) = a\frac{v - v_{min}}{v_{max} - v_{min}} - b_{collision} \tag{2}$$

where v,v_{max}, v_{min} are the current, minimum and maximum speed of the ego-vehicle respectively, and are two coefficients.

V. Evaluation Metrics

Deep reinforcement learning algorithms are considerably sensitive to implementation details, hyper-parameters, choice of environments, and even random seeds. The variability in the execution can put reproducibility at stake. So it is necessary to have common evaluation metrics across all the algorithms. Deep reinforcement learning algorithms are of two types value-based and policy-based. It is difficult to compare value-based algorithms with policy-based algorithms because both work differently with different tuning parameters. In this section, we are going to define the base of evaluation metrics for value-based and policy-based methods.

A. Highway Environment Configuration

Common configuration parameters for the environment are necessary for a fair evaluation. This section defines the common configuration parameters which are used across the value-based and policy-based algorithms.

centering position	[0.3, 0.5]
collision reward	-30
duration	30
lane change reward	-1
lanes count	4
policy frequency	5
reward speed range	[20, 40]
right lane reward	1
simulation frequency	15
vehicles count	20
vehicles density	2

B. Highway Environment Configuration For Value-Based Algorithm

Following are the additional configuration parameters for value-based algorithms.

action type	DiscreteMetaAction
observation shape	[600, 150]
stack size	4
observation type	GrayscaleObservation
RGB weights	[0.2989, 0.5870, 0.1140]

C. Highway Environment Configuration For Policy-Based Algorithm

Following are the additional configuration parameters for policy-based algorithms.

action type	ContinuousAction
observation features	presence,x,y,vx,vy,cosh,sinh
observation features range	x: [-100,100], y: [-100,100], vx: [-20,20], vy: [-20,20]

D. Evaluation

In order to evaluate different algorithms we trained all algorithms with 5000 epochs and compared the plots of mean reward versus the number of episodes. The plot that has faster learning and more mean reward is better.

VI. Experiments

To train the agent on each algorithm we used the GPUs from Turing cluster with 4 cores, 64GB ram and one A100 GPU. We iterated over different hyperparameters and settled on the following hyperparameters for each algorithm,

A. DQN

epsilon start	1.0
epsilon end	0.05
epsilon decay	10000
gamma	0.99
batch size	32
buffer size	10000
learning rate	0.0001
target update frequency	2000
total episodes	5000

B. DDDQN

epsilon start	1.0
epsilon end	0.05
epsilon decay	10000
gamma	0.99
batch size	32
buffer size	10000
learning rate	0.0001
target update frequency	2000
total episodes	5000

C. DDPG

gamma	0.99
batch size	64
buffer size	10000
learning rate actor	0.0001
learning rate critic	0.001
tau	0.001
target update frequency	2500
total episodes	5000

VII. Results

From the plots in the report, it can be observed that DQN learned at 2000 episodes and had a mean score of 90 after it learned. While DDDQN learned at 1300 episodes and had a mean score of 130. This aligns with the hypothesis that DDDQN is an improvement over DQN. DDPG was trained on continuous action space so it took time for the agent to follow the lane. DDPG learned at 1200 episodes and had a mean score of 230 but we can't compare it with other algorithms because it was trained on continuous action space. PPO learned at 800 episodes and had a mean score of 180 which is much higher than DQN and DDDQN.

VIII. Conclusion

After training value-based and policy-based Deep Reinforcement Learning algorithms like DQN, DDDQN, PPO, and DDPG respelanesely in a highway environment, it can be concluded that PPO with discrete action-space gave us better

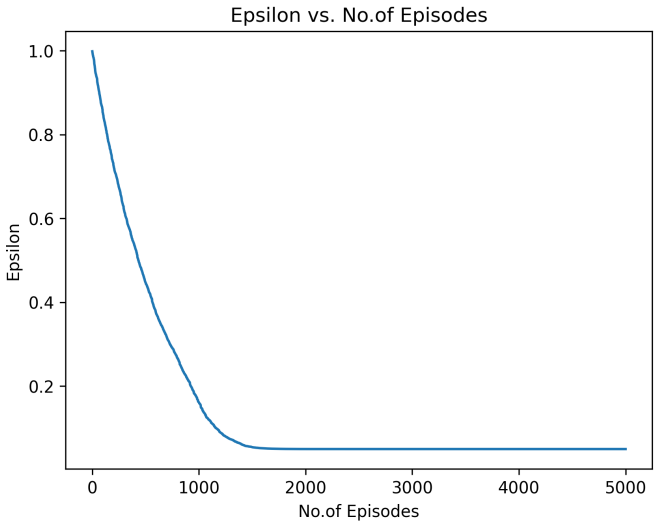

Fig. 8. DQN Epsilon vs No. of Episodes

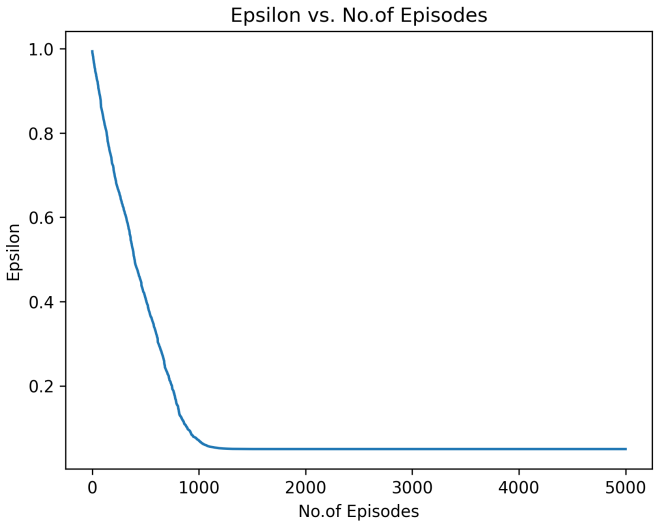

Fig. 10. DDDQN Epsilon vs No. of Episodes

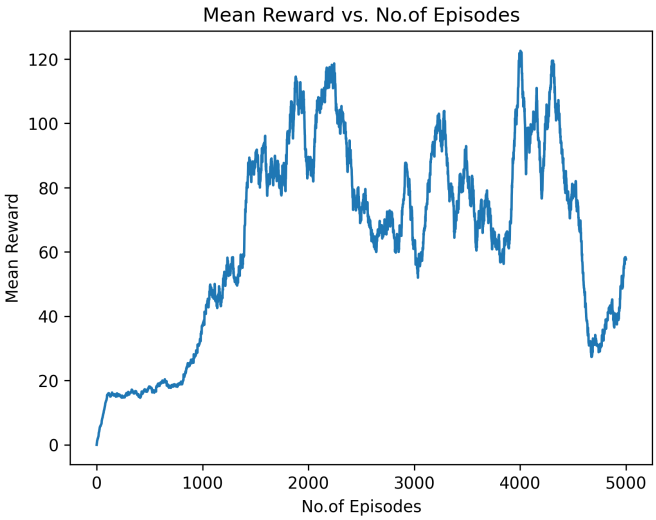

Fig. 9. DQN Mean Reward vs No. of Episodes

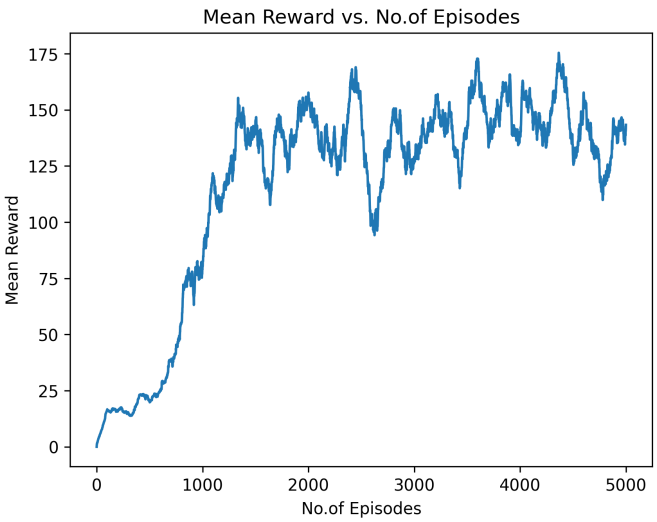

Fig. 11. DDDQN Mean Reward vs No. of Episodes

reward and good performance on testing. PPO agent learned to change lanes without colliding with other agents in less number of episodes compared to DQN, DDDQN, and DDPG. DDPG with continuous action space resulted in taking a lot of iterations to train an agent on continuous action space. For the future, we wish to improve the decision-making performance of an agent in continuous action space as real autonomous vehicles works in continuous action space.

REFERENCES

[1] E. Leurent, "An environment for autonomous driving decision-making," https://github.com/eleurent/highway-env, 2018.

[2] Y. T. Quek, L. L. Koh, N. T. Koh, W. A. Tso, and W. L. Woo, "Deep q-network implementation for simulated autonomous vehicle control," *IET Intelligent Transport Systems*, vol. 15, no. 7, pp. 875–885, 2021. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/itr2.12067

[3] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *CoRR*, vol. abs/1509.06461, 2015. [Online]. Available: http://arxiv.org/abs/1509.06461

[4] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. K. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *CoRR*, vol. abs/2002.00444, 2020. [Online]. Available: https://arxiv.org/abs/2002.00444

[5] X. Tang, B. Huang, T. Liu, and X. Lin, "Highway decision-making and motion planning for autonomous driving via soft actor-critic," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 4706–4717, 2022.

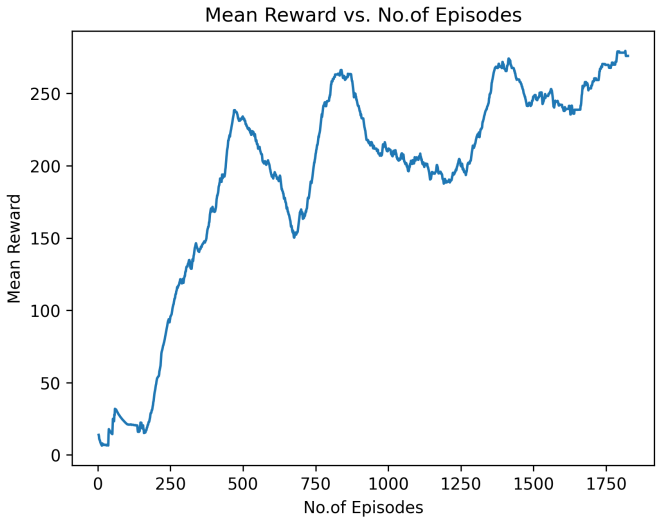

Fig. 12. DDPG Mean Reward vs No. of Episodes

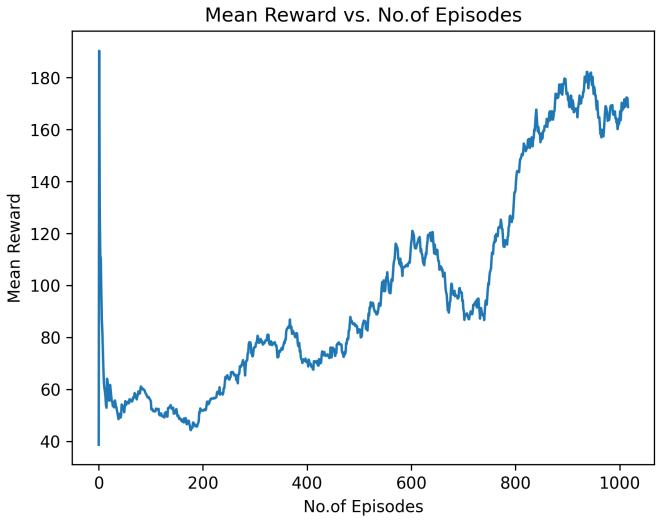

Fig. 13. PPO Mean Reward vs No. of Episodes

[6] E. Leurent and J. Mercat, "Social attention for autonomous decision-making in dense traffic," *arXiv preprint arXiv:1911.12250*, 2019.

[7] A. Rana and A. Malhi, "Building safer autonomous agents by leveraging risky driving behavior knowledge," in *2021 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI)*. IEEE, 2021, pp. 1–6.