INM707 DEEP REINFORCEMNT LEARNING

COURSEWORK

Vivek Kanna Jayaprakash
Student No : 210033212
MSc Data Science
City University of London
London, United Kingdom
Email : Vivek.Jayaprakash@city.ac.uk

I. ENVIROMENT AND PROBLEM TO BE SOLVED

Problem:In the world of today, E Commerce has rapidly occupied our lives and completely changed the way we shop. Be it a small paper clip of the latest iPhone our first thought is to look at the various popular E commerce websites for the prices and ordering it online rather than spending time in queues at one's local retailer. Since this rapid demand for supply for retail goods skyrocketed the surge in the services which makes the sales and delivery happen did too.

Advent of AI in the last few years has taken over the business and financial decision of every industry and E commerce is no exception.The main problem a user faces during the entire order – delivery process is late delivery. We have all been there when we ordered and items and it has taken more than the usual time to reach us. This is majorly caused due to the warehouse delivery mishap caused due to human labour.[1]

In this project we are attempting to use reinforcement learning techniques to teach automatic warehouse robot within an E-commerce website company's warehouse to pick items and deliver the same to the inputted goal or designated delivery location in a maze.

We use Deep reinforcement learning techniques to find the shortest path for the robots to pick up the goods from the required square and deliver them to their desired packing location in the same warehouse so the items can be packed and shipped.

For increasing the efficiency and to maximise the productivity we attempt to identify the path which is the shortest distance from the packing area to all other area of the shelves where the items are located.

We consider this set up as a Maze atmosphere where the delivery robots needs to navigate through a series of squared checks to reach a goal. They should do the same by avoiding the black squared boxes thereby the terminal states as shown in figure 1. We Modelling our scenarios based on the paper published by Dreyfus, G., 2005 [2] *Mahrukh in 2019 [4]* We are combining both the paper to experiment by changing the terminal states and all parametric values.

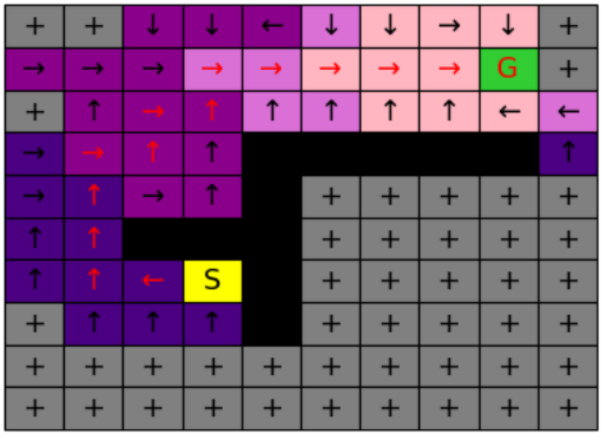

Figure 1: Maze environment

From figure 1 we can identify that the S is the starting point and G is the goal and the Red arrows indicate the shortest path to reach the goal.

For this question we implement a deep learning and Q learning approach to solving our query. Epsilon greedy is considered as the policy for finding the shortest path required for the robots to pick up a good from a said location within the warehouse and deliver it to the desired goal area in the MAZE environment.

Environment : As discussed We use the Maze environment to find the shortest path required for a warehouse robot to reach its said goal

The MazeEnv consists of the CoreEnvs as the Gym style environment in a totally understandable and reliable format. It does the same by utilising the interfaces or the other period of mappings from the MazeState to the periodical observations and from the MazeAction to the Said key action.

The implantation of this ENV is simple as from maze import MazeENV and with the said command the cells will be ready to perform the initial set of training run.

The rewards and punishments are inbuilt as per every environment for this set of cells.

II. STATE TRANSISTION FUCTION AND REWARD FUCNTION

State Transition :

For this problem the agents will interact with the environment on a continues basis and keeps on its process of knowledge adaptation for the entire maze and also updates the knowledge of the policy. And thereby reaches a goal of achieving the maximum reward.

The path to reach the goal is the learning curve in which the agent aims to achieve and also making sure it doesn't crash into the said obstacles. The optimal path is identified by the entering point to the goal by continuously maximising the reward set up and minimising the punishments set up.

We attempt to formulate the Problem by modelling the environment as a stochastic finite state machine various said actions send from the agent. And give out outputs as observation and reward sent to the agent's as follows. The environment is then modelled

The environment consist of states, actions and rewards in which the states and actions are the considered input for the Q learning atmosphere

In this problem the states are the entire blocks of the maze, there are four possible action (left, right, down and up) The agent can take any significant action to reach the goal provides it the shortest path and it doesn't crash in tot eh terminal back squared chequered block.

Reward function:

In our experiment the reward function is not identified at the beginning but the agent will identify the same once it moves from one block to another on the set of scale. Thereby the required function clearly fully explores and covers the environment for learning the reward.

The black boxes are all terminal states having (minus) -100 as reward in which we teach to agent to not crash into them.

The non-black boxes are set with reward as -1 in this case as our goal is to minimise the punishments and to identify the shortest way to the desired goal box. The reward is set to be -1 and not 1 because the agent should not keep rotating and travelling up in them boxes thereby increasing its rewards.

The rewards, punishments are set up automatically by the said maze environment.

The maximum cumulative reward thereby minimizing its cumulative punishments makes the agent need to find the shortest path to the packaging goal area are where the reward is set to 100

$R(s) = R(s) + \text{alpha}\ (r - R(s))$

Alpha = learning rate

R = immediate reward.

This type of immediate learning is required because it uses the learning has a serious effect on it next action.

III. Q LEARNING PARAMETERS AND POLIFY

The main formula for q learning also knowns as the bellman's equation is signified as :

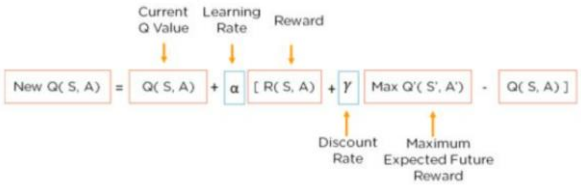

Figure 2: Bellman's equation

Where

Q() = Q learning algorithm

S is the previous state

A is the previous action

S' = current state

α – alpha or learning rate where learning rate (0-1) is the rate of much in which the state accept the new value in continuation of the old value. Takes in previous q values and movies it to the upside

ϒ = gamma or discount factor (0-1) is used to get a good balance between the immediate and future reward. We can only apply discount to the future reward.

Policy - Epsilon greedy function :

Epsilon greedy is very early iterative method to balance exploration and exploitation by choosing between them randomly.

Exploration make the agent improve the current knowledge on every said action to make sure the long view goal advantage meanwhile the exploitation being the greedier one chooses the best action using the states current action value estimates.

The agent will always choose the random state until otherwise the greedy action is greater than its parametric epsilon.

IV. RUN THE Q LEARNING ALGORITHM

For the training we input the said values

No of episodes = 300

epsilon = 0.1 # The percentage of the timeframe were we take the best action. We are choosing the low number so that agent explores the entire training phase.

Discount factor or gamma = 0.9 We set the value to the maximum so that we get a maximum reward.

Learning rate or alpha = 0.9. We experiment with higher alpha.

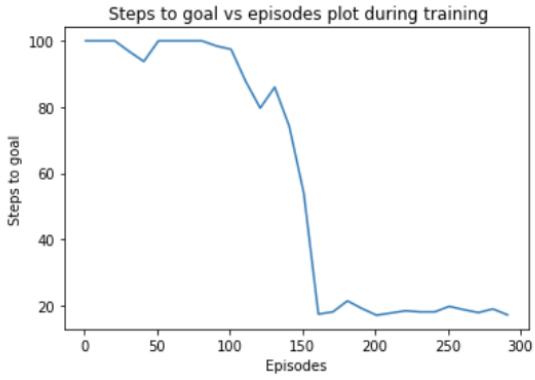

Figure 3: Steps to goal vs episodes during training

From figure 3 We see that with the increase in the number of episodes there is decrease in the steps to reach the goal. Thereby understanding that the model is learning as the episodes process taking over its learn reward and using it as the learning for the next step of action and thereby finding the shortest path.

From the same we see that it reaches the goal at it 300 episodes at less than 20 steps.

In the next step After we set up a different goal saying the goal is at 7,6

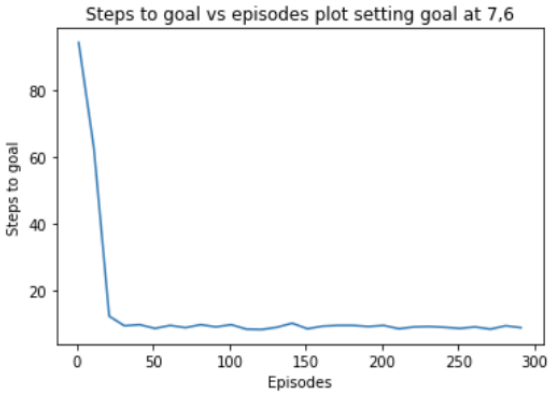

Figure 4: Steps to goal vs episodes setting goal at 7,6

we understand that the steps to reach the goal is further decreased mainly to its earlier learnings also since the goal is significantly closer than the training set.

Setting up a tougher goal:

We set the goal to 1,6

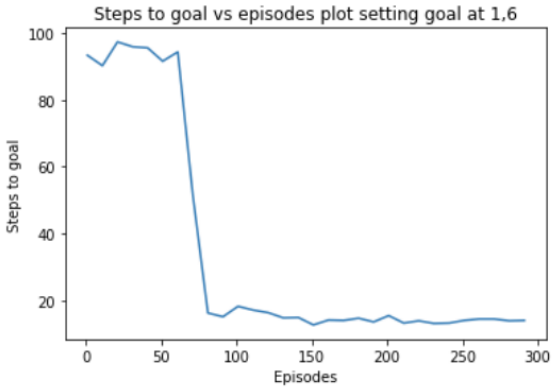

Figure 5: Steps to goal vs episodes setting goal at 1,6

In this for the figure 5 we understand that the time taken to reach is little longer than the previous goal as because of the longer path to reach the goal.

V. REPEAT THE EXPERIEMNTS WITH DIFFERENT PARAMETER AND POLCIES.

Changing the policy value.

(i)Epsilon value is changes to [.01] ,[0.5],[0.9]

We know that increasing the epsilon values significantly make the algorithm greedy and makes the policy learn quicker by not so viable in the long run.

Lets put the state to test by giving three significant epsilon values to experiment with so we can identify which path does the best.

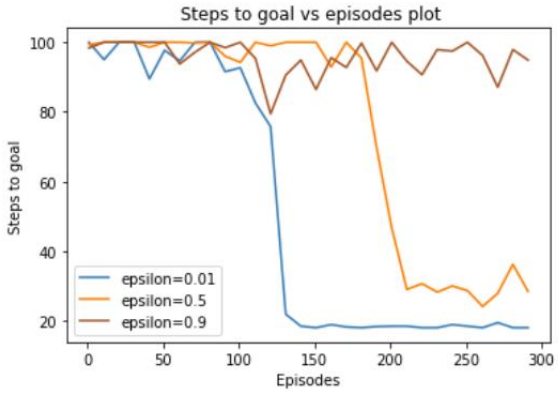

Figure 6 : steps to goal vs episodes based on epsilon

We see the agent takes comparatively very fewer iterations to reach the best solution for the high value of epsilon. As discussed earlier this is due to the exploitation greedy behaviour.

However a very small or large epsilon values may or may not converge or may take a long time to converge thus the only advantage of using a larger value of epsilon is that it learns quicker to learn.

In contrast the benefit of having a lower epsilon rates are that they will always in the most optimal solution as sown in the graph.

(ii)Alpha value or the learning rate:

Alpha value is changed to [0.01] ,[0.5],[0.8]

The higher amount of alpha signifies the learning can be occur at a faster phase

(iii) Gamma value or discount factor

Gamma value is changed to [.01] ,[0.4],[0.8]

Setting the gamma to lower amount will make the reward to stand out continuing to the current rewards and gamma value higher will consider the rewards to be significantly good in the lingered term considering the strive to operating a higher amount of reward in the longer run.

VI. Analyse the results quantitatively and qualitatively

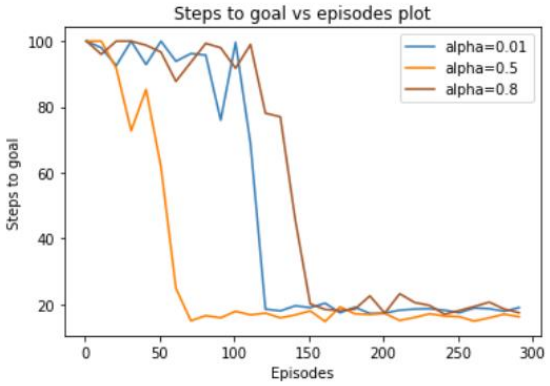

Figure 7 : Steps to goal vs Episodes based on different alpha

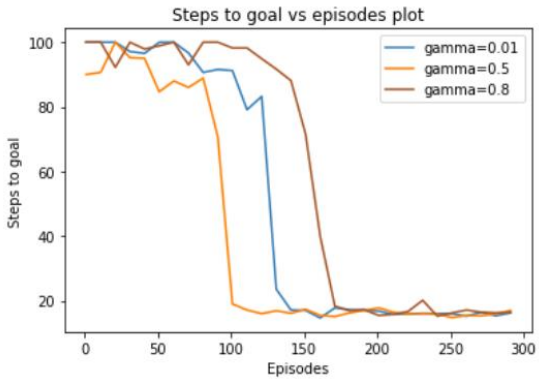

Figure 8 : Steps to goal vs Episodes based on different gamma

Based on figure 6, 7 and 8 we can clearly understand that our experiment follows the literature. Increase in epsilon reduces the steps taken to reach the goal while the gamma value increase is significantly good rewards. The alpha phenomenon need to be noted that the lowers and higher value of alpha converges later while alpha = 0.5 converges earlier which suggests better scenario.

Advanced tasks:

VII. Implementing DQN with two improvements

Enviroment and Problem to be solved

GYM environment is put to use in this scenario and use pf the cartpole game is made to implement the model

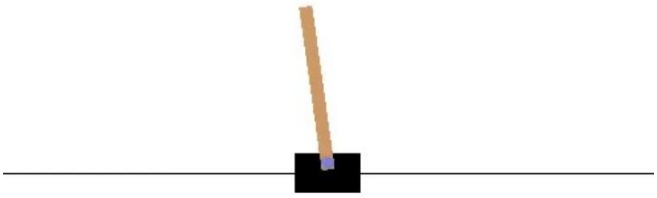

Figure 9: Cartpole game from gym

(i)DQN : In simpler words DQN replaces the Q table with a neural network. Instead of mapping a state action pair to a q-value a neural maps is put to action and q value pairs
It uses 2 neural networks which have same architecture and different weights. With every N steps the writing form the main network are copied to the target area.

Considering the said both networks is done it thereby leads to a more stable learning processes and learns the algorithm effectively.

The main network wrights replace the target network weights every 100 steps.

In this scenario we are using the Cartpole problem and implement DQN

We are combing three different paper to do the same cart pole experiment in which each have chosen to do DQN, DDQN and Duelling separately and have chosen the same type of approach by various factors and we shall experiment with the same [3], [100, [11]

```
# Hyper Parameters
BATCH_SIZE = 32
LR = 0.01  # learning rate
EPSILON = 0.9  # greedy policy
GAMMA = 0.9  # reward discount
TARGET_REPLACE_ITER = 100  # target updat
e frequency
MEMORY_CAPACITY = 2000
```

Said executing takes a very long time to run and interpolate the results.

```
Episode  1199
Ep:  1199 | Ep_r:  606.61
```

For two significant improvement we use Double Deep Q networks (DDQN) and Duelling

The main reason for choosing the two said techniques is mainly because it is popular among all other algorithms. A trial was made for prioritised reply algorithm but unfortunately the notebook crashed due to insignificant storage so had to resort to the two popular techniques

(ii)Double DQN: It utilises two identical neural networks models in which the first one leans during the process of the experience reply which is just a mere copy of the final episode in the initial model. The q value generated is the actual model. The q values is therefore computed with the second model Each neuron in the output will increase and increase till the difference between each output values is high.

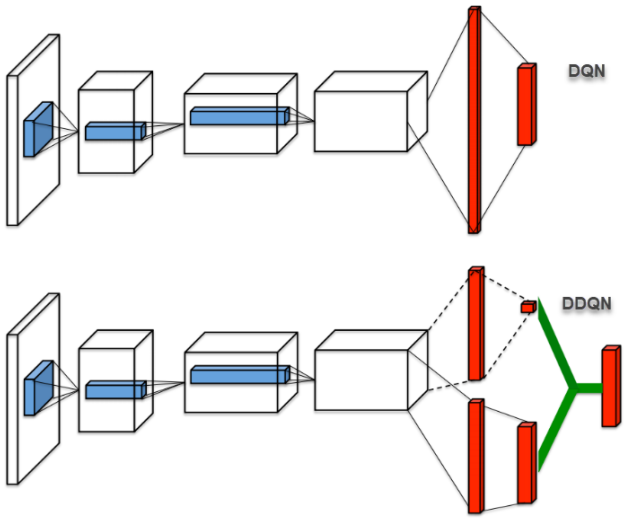

Figure 10: DQN VS DDQN

Using the second model which is the copy of the main model of the last episode since the value different of the second model is lower than the main model we use the second model to attain q value therefore the process continues.

(iii) Duelling DQN -The structure in the model is the difference between the Duelling and DDQN.

The forumla is Q(s,a) = V(s) + A(S.a)

V(s) is the value of the states (s)and A is the advantage of theaction (a)

This value is independent of the action.To understand how good it is to be in a particular state.

Explain in the image

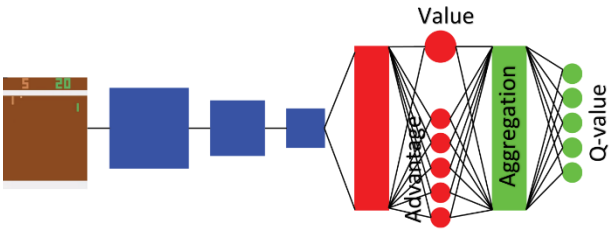

Figure 11: DUELLING

The output of the model is the state value inclusive of the advantage of action but for training we use the same value for the targes

VIII. Analyse the results Qualitatives and quantitatively:

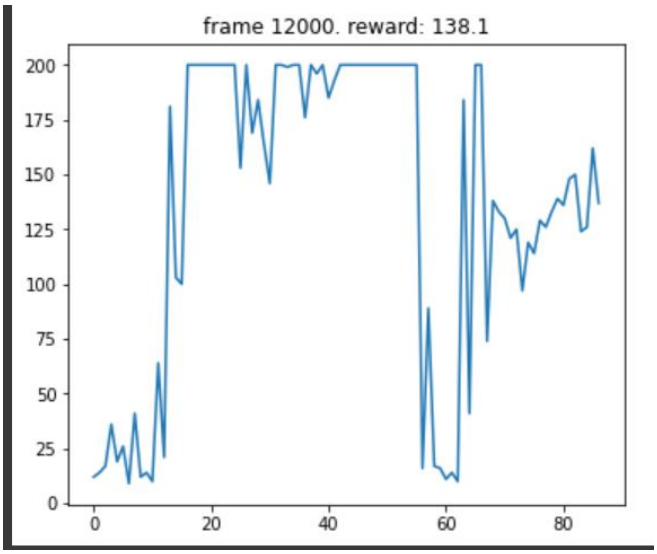

Figure 12 : DDQN Rewards

Figure 12 explains The DDQN reward form a exponential path to reach the ward of 138 after the set of episodes run
The peak in the data set at 80 explained with the figure below

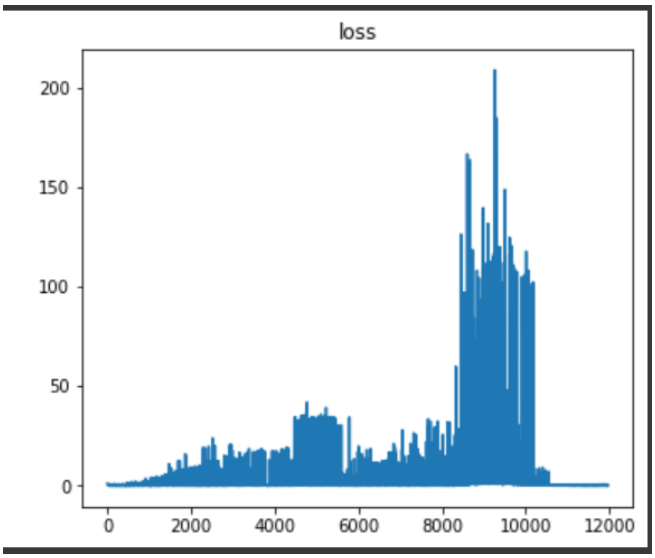

Figure 13: Figure 12 : DDQN Losses

Figure 13 explain The loss is very heavily around the 80 therefor for the loss in reward at the end of episode 12000 loss is very heavily minimalized

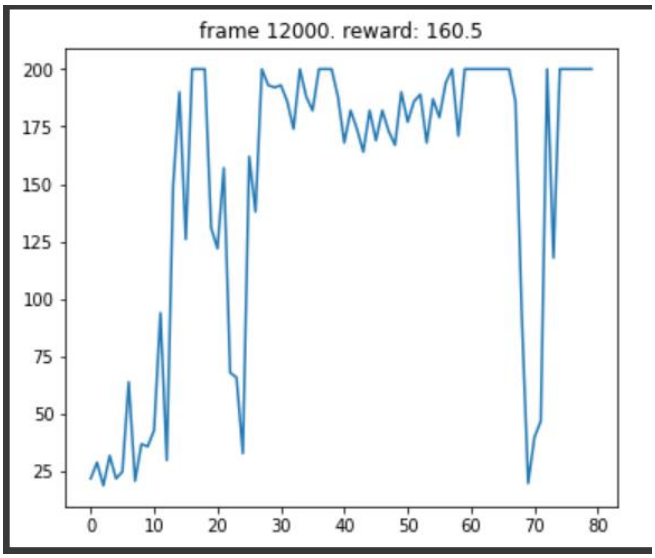

Figure 14: Duelling rewards

Figure 14 explains The Duelling reward form a exponential path to reach the ward of 160 after the set of episodes run
The peak in the data set at 80 is similar to the DDQN but it continues till the almost very end. It is explain with the figure below

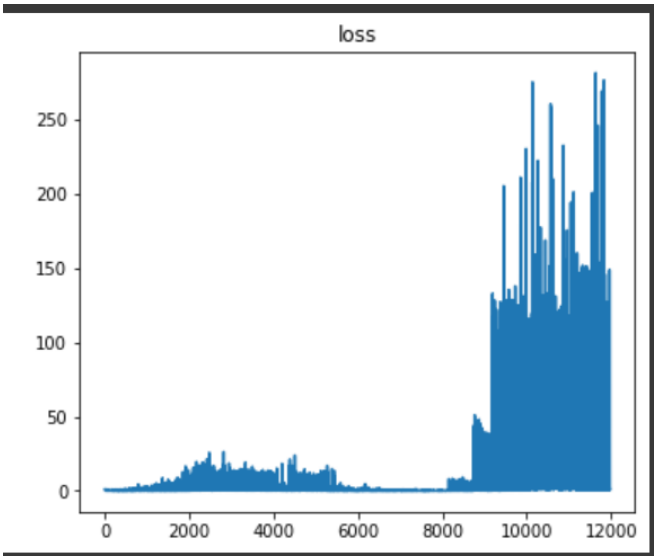

Figure 15: Duelling losses

Figure 13 explain The loss is very heavily around the 80 but significantly continues till the end but the end of episode 12000 loss is very heavily minimalized
Therefore from the above performed experiments we can conclude that the duelling performs better than DQN and DDQN.

IX. IMPLEMENTING RL ALGORITM FROM RLLIB IN ATARI LEARNING ENVIORUMENT.

Atari is a third party owned game maker company by Nintendo Japan which collaborated with the Arcade learning environment (ALE) to creating Atari environment which is significantly used fin the field of reinforcement learning for many contribution to produce heaps of games being developed on the reinforcement learning teaching techniques.

It has object oriented framework which supports to add agents and agents into the system and has d emulation core uncoupled from rending and sound generation

The programmer can easily extract games based on commands and experiment with over 100 ATARI 2600 games. Python support and Native Open AI gym support are significant features.

In this experiment we use the RL Lib

RL LIB is a open source library which supports and offers production level Reinforcement learning wrhile managing a very user friendly API. The amount of convolution layer building is not needed in RL LIB as the Algorithms are already pre coded into the system.

Algorithm : Even though there are several different algorithms such as PPO |SAC|Impala present on RL Lib we chose DQN because it is very familiar and easy and the easier solution is always the best one.

We are utilising the Ray Rl lib library coding for the entire dataset to make the experiment and we are using the space invaders games for the scenarios[1],[12]

We use the space invaders game in the Atari learning environment to analyse the results based on the said calculations.

Due to the limited run time on the google collab GPU the code as follows

```
config = {
        "env" : "SpaceInvaders-ram-v0",
        "dueling" : tune.grid_search([True,False]),
        "double_q" : tune.grid_search([True,False]),
        "lr" : tune.grid_search([0.1 , .3, 0.5]),
    }
```

For creating the DQN and duelling model including changing the Learning rate keeps on failing due to the limited hence we have performed the task on a single DQN model

X. ANALYSE THE RESULT QUALITATIVELY AND QUANTITATIVELY:.

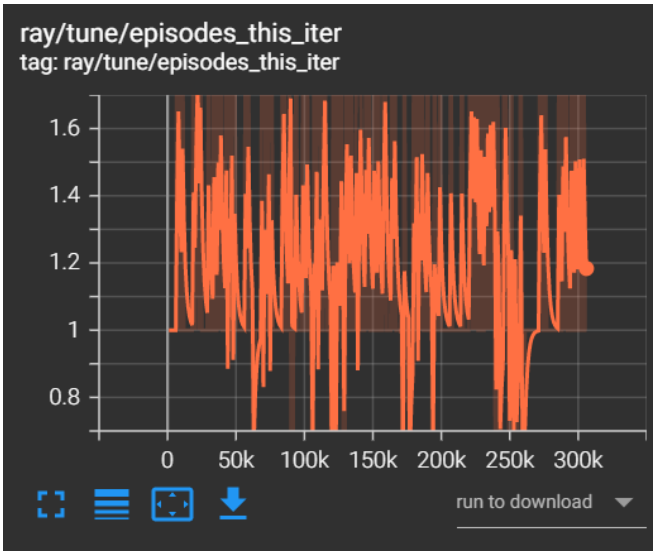

Figure 16 : Episodes per Iteration

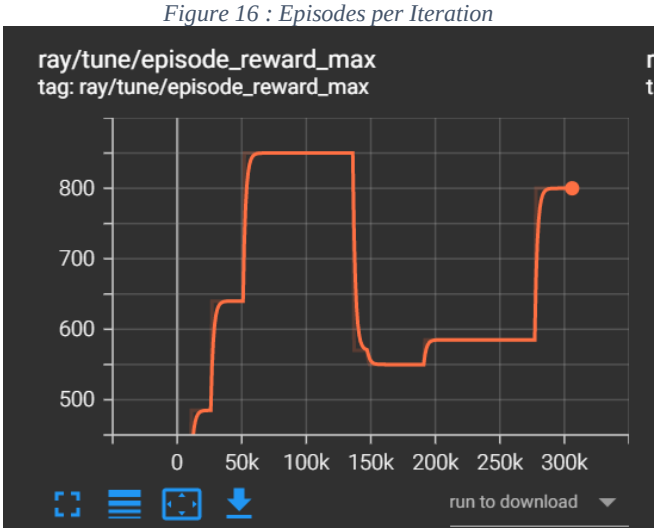

Figure 17 : Episodes per Max reward

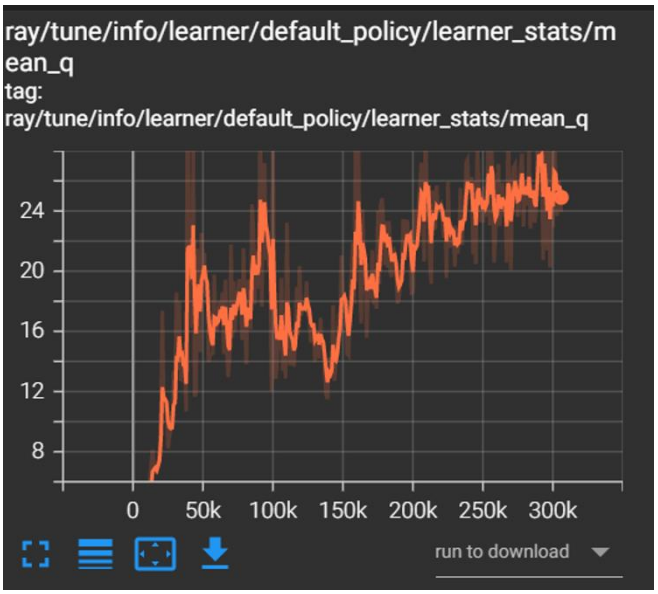

Figure 18: Mean Q

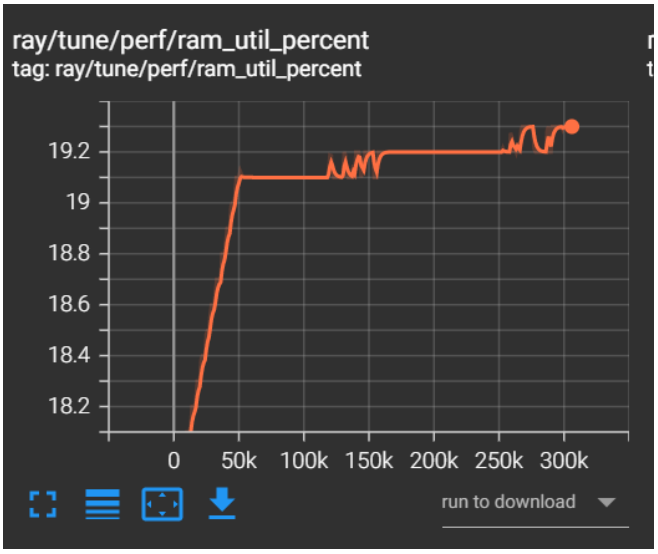

Figure 19: Ram Utilisation

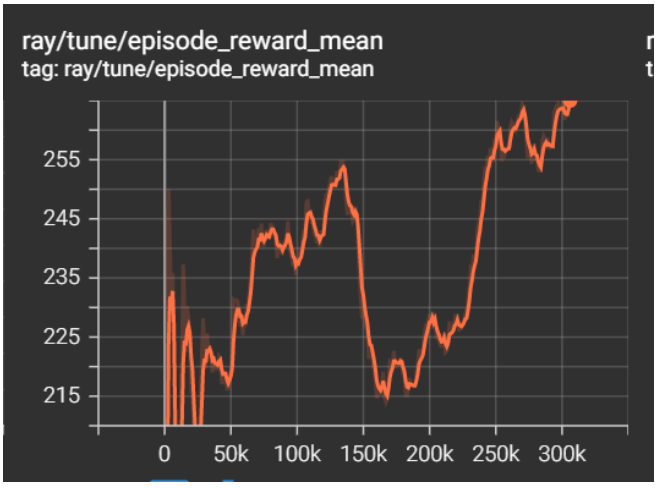

Figure 20; : Episodes per mean reward

ame using TensorFlow just to make changes to the pathAll the above mentioned chats prove that the at 300 episodes the experiment started working and the ram utlised has goen tot maximum at 50K iteration.

The model run a dnspace invaders can be experimented likely n
If the google runtine had been present we could have been able to analyse different algoithms abnd learning and this is the next significant imrpoveentmrewuired on this project.

XI. Implementation of PPO

PPO is also another set of algorithms which is policy gradient method where the policy is updated explicitly.
We implement PPO on the Gym environment on the cartpole game using TensorFlow just to make changes to the path.

References

[1]. Docs.ray.io. 2022. *RLlib: Industry-Grade Reinforcement Learning — Ray 1.12.0*. [online] Available at: <https://docs.ray.io/en/latest/rllib/index.html> [Accessed 24 April 2022].

[2]. Dreyfus, G., 2005. *Neural networks*. Berlin: Springer

[3]. GitHub. 2022. *GitHub - Mahrukh-Niazi/train_neural_network_TFD: Train a neural network on a subset of the Toronto Faces Dataset (TFD) and optimize it.*. [online] Available at: <https://github.com/Mahrukh-Niazi/train_neural_network_TFD> [Accessed 24 April 2022].

[4]. GitHub. 2022. *GitHub - Mahrukh-Niazi/train_neural_network_TFD: Train a neural network on a subset of the Toronto Faces Dataset (TFD) and optimize it.*. [online] Available at: https://github.com/Mahrukh-Niazi/train_neural_network_TFD> [Accessed 24 April 2022].

[5]. GitHub. 2022. *GitHub - shangeth/Lunar-Lander-Deep-RL: Deep Q network, DDQN, Dueling DQN for Lunar Lander Environment.*. [online] Available at: <https://github.com/ nar-Lander-Deep-RL> [Accessed 24 April 2022].

[6]. Khosrow-Pour, M., 2002. *Web-based instructional learning*. Hershey, PA: IRM Press.

[7]. Medium. 2022. *Deep Reinforcement learning: DQN, Double DQN, Dueling DQN, Noisy DQN and DQN with Prioritized…*. [online] Available at: <https://medium.com/@parsa_h_m/deep-reinforcement-learning-dqn-double-dqn-dueling-dqn-noisy-dqn-and-dqn-with-prioritized-551f621a9823> [Accessed 24 April 2022].

[8]. QLearning, A. and elysium, d., 2022. *Alpha and Gamma parameters in QLearning*. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/1854659/alpha-and-gamma-parameters-in-qlearning> [Accessed 24 April 2022].

[9]. Sutton, R. and Barto, A., 2014. *Reinforcement learning*. pp.32-36.

[10]. vJingci, L., 2022. *GitHub - LyapunovJingci/Warehouse_Robot_Path_Planning: A multi agent path planning solution under a warehouse scenario using Q learning and transfer learning.*. [online] GitHub. Available at: <https://github.com/LyapunovJingci/Warehouse_Robot_Path_Planning> [Accessed 24 April 2022].

[11]. Whiteson, S., 2010. *Adaptive representations for reinforcement learning*. Berlin: Springer-Verlag.

[12]. GitHub. 2022. *GitHub - 4dot4/AtariBreakout: Atari Breakout game made in c and raylib*. [online] Available at: <https://github.com/4dot4/AtariBreakout> [Accessed 24 April 2022].

[13]. GitHub. 2022. *GitHub - google-research/batch-ppo: Efficient Batched Reinforcement Learning in TensorFlow*. [online] Available at: <https://github.com/google-research/batch-ppo> [Accessed 24 April 2022].