Comparison Study Of Reinforcement Learning Algorithms

1st Anubhav Shrimal
CSE (M.Tech)
MT18033
IIIT Delhi
anubhav18033@iiitd.ac.in

2nd Mahika Wason
CSAM (B.Tech)
2016241
IIIT Delhi
mahika16241@iiitd.ac.in

3rd Rupal Jain
CSE (B.Tech)
2015081
IIIT Delhi
rupal15081@iiitd.ac.in

***Abstract*—OpenAI Gym is a toolkit used for a purpose of developing and comparing reinforcement learning algorithms. It has a collection of environments that we can use to work out our reinforcement learning algorithms. In our project, we worked on Pacman and Frozen Lake environments. Pacman agent has an objective to maximize its award points by eating the big fruits, while keeping itself safe from the ghost. Frozen Lake agent has an objective to reach the goal state while being alive. If it steps into a hole, it gets a -1 reward and dies. Classical AI agents require you to have intact knowledge of the rules and the states of the game. In this project, we implement Q-Learning approach on these two environments. This doesnt require us to have a complete knowledge of the states and rules beforehand, the algorithm learns it. We also implemented the Deep Reinforcement Learning (DQN) approach and drew comparisons from the results obtained from the two.**

***Index Terms*—Reinforcement Learning, Q learning, Pacman, Deep Q Network**

I. Litreature Review

As a part of his PhD thesis, Watkins, J.C.H. [1] (1989) namely, Learning from Delayed Rewards, introduced a simple and intuitive approach for agents to learn by mistake for how to act optimally in controlled Markovian environment. It makes extensive use of the Dynamic Programming paradigm which works by successively improving the evaluations of the quality of particular actions at particular states. The Q-Learning approach introduced by them could learn the optimal control directly without the need to model the transition probabilities or expected rewards of the MDP.

DeepMind technologies, in their 2013 paper on Deep Reinforcement Learning [2], introduce the concept of approximating the Q-Value function for large state space environments. In this approach, they used Convolutional Neural Networks with fully connected layers. Their input is raw pixels and output is a value function which helps in determining the future rewards. They applied their method on seven Atari games from the Arcade Learning environment. In the paper [3] performs a comparison study on the performance of different activation functions for convolutional neural network. It concluded that Exponential Linear Units (ELUs) performs best in comparison to other activation functions. They argues that ELUs speeds up the learning process in the Deep Neural Networks and leads to higher classification accuracy. In our project, we have incorporated these techniques in our project to observe how they perform on Pacman and Frozen Lake environments by tuning three hyper parameters namely, decay of exploration rate, learning rate, and discount factor. And, we present our analysis and draw conclusions from the plots obtained.

II. Algorithms

In Reinforcement Learning algorithms, we try to match an input to an action to maximize the rewards or profit. System is not fed with hard-coded values to determine which actions to take, rather it follows the methodology of Trial and error and learns the best actions itself. It tries to take future performance into context in an attempt to perform better, because this approach is designed in a way that its present actions affect its future rewards.

A. Q learning

It is an off-policy TD control algorithm in which we are not concerned with the state values, instead we care about the state-action value pair the impact of an action on a state.

B. Q table

A Q-Table (Q for quality) is where we calculate the maximum expected future reward for each action at each state. The columns are all the actions possible from a state, rows are all the states. It helps us determine what our next best action is. In Q-Learning, we dont implement a policy, per se. We just update our Q-Table to always choose the best action.
Parameters:

$$Q(s,a) = Q(s,a) + \alpha\left(r + \gamma Q_{max}(s',a') - Q(s,a)\right) \quad (1)$$

- Q(s,a): Q-Value at state we were previously.
- Alpha: Learning rate
- Gamma: discount factor
- Q(s, a): Q-Value of the state s which we reach after taking the action a.
- r: reward
- Max Q(s, a): Value of action a for which the Q-Value at s is maximum.

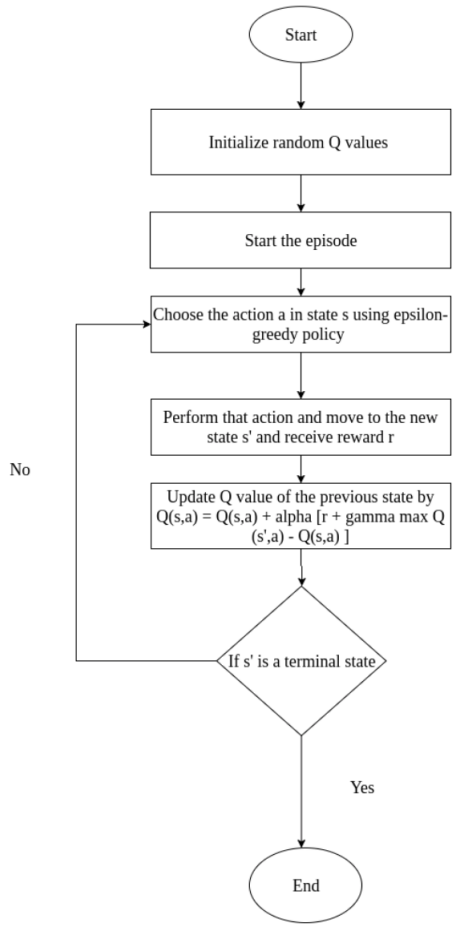

Fig. 1. Flow Diagram of Q learning

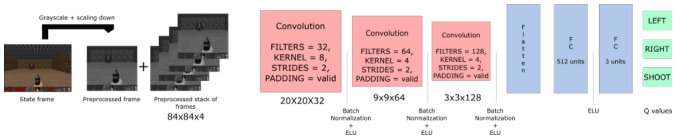

Fig. 2. DQN Architecture

C. Deep Q Networks

This approach is mainly used to deal with environments having large state space environments. Q-Learning is not practically feasible on such environments as the Q-Table will become infinitely large and iterating over such a table would be computationally very expensive. Hence, the best approach would be to create a neural network that will approximate the different Q-Values for each action for a given state. The Loss is calculated and the gradient is taken w.r.t the weights to update the Q network:

$$\Delta w = \alpha \left[\left(r + \gamma \hat{Q_{max_a}}(s', a, w) \right) - \hat{Q}(s, a, w) \right] \nabla_w \hat{Q}(s, a, w) \tag{2}$$

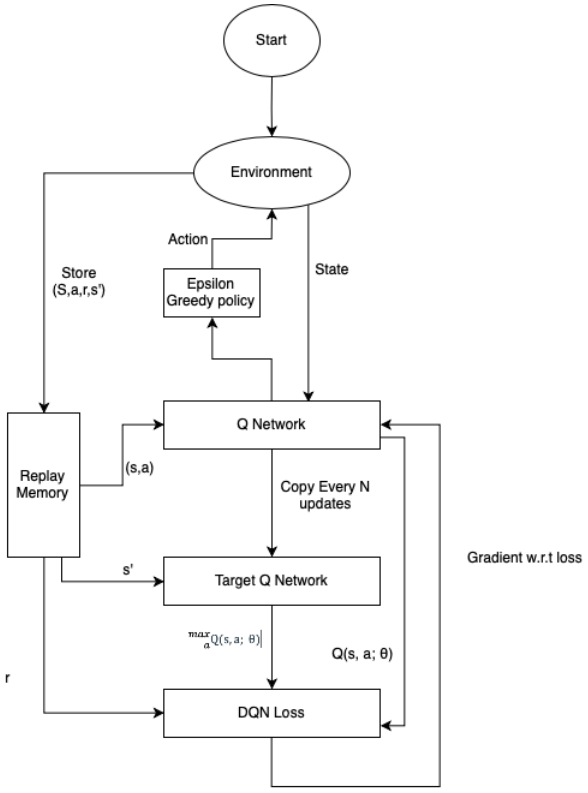

Fig. 3. Flow Diagram of Deep Q Learning

III. RESULTS

TABLE I
HYPERPARAMETERS TUNING

Number of Episodes	10000	15000	20000
Average Reward	0.472	0.48	0.4897
Learning Rate	0.8	0.9	0.6
Average Reward	0.472	0.4584	0.5072
Discount Rate(gamma)	0.95	0.75	0.99
Average Reward	0.472	0.377	0.4692
Decay Rate(epsilon)	0.005	0.008	0.003
Average Reward	0.472	0.4822	0.466

IV. PRACTICES NOT TO FOLLOW

- Large Learning rates heavily updates the model weights at each step this makes the model forget the previously learned weights fast and also generalizes less.
- Exploration rate must be significant enough so as to enable the agent to explore, however if the exploration rate is too much even after the agent is trained to some degree, it can result in the agent ignoring it's learning so far.
- While we can observe certain patterns, it is erroneous to make strong inferences based on limited training of the agent. A good amount of training is thus inevitable.

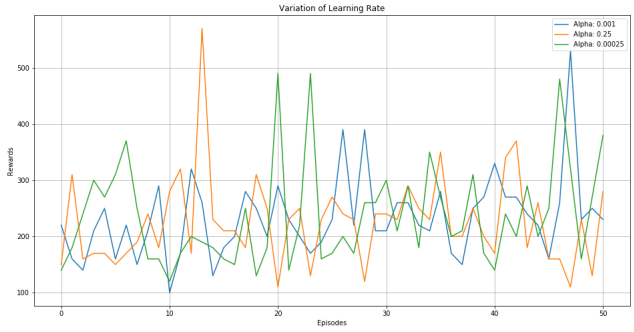

Fig. 4. The green line in the above graph is seen to show consistent rewards over the training period. The orange line, on the other hand is portraying sudden peaks and troughs. This shows that high learning rates leads to adverse changes in the model.

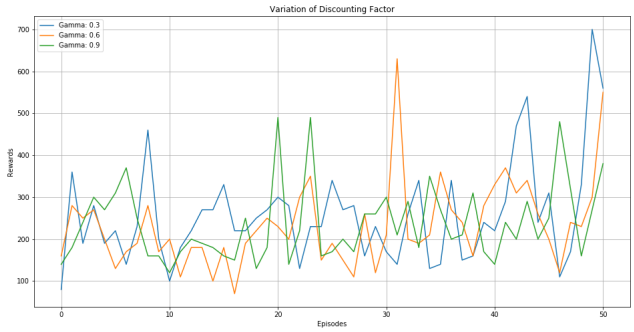

Fig. 5. Discounting rate decides the importance of future rewards. The green line shows the best performance as compared to the other two. This is because this enables the agent to learn more.

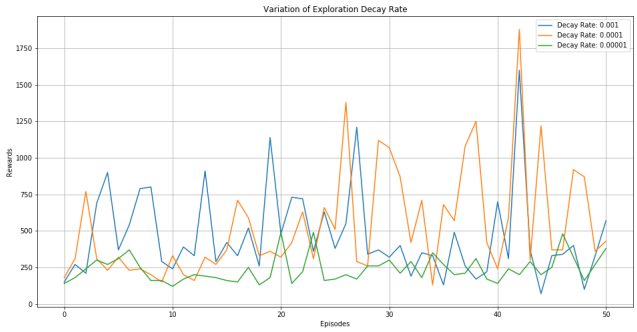

Fig. 6. Decay rate defines the degree of exploration by an agent. The orange line corresponding to an intermediate value of decay rate. It performs better as compared to the blue and green lines. This is simply because the agent must use it's previous learning as well as continue to learn. Extreme of either strategy results in low performance by the agent.

V. Conclusion

- We conclude that due to large space requirements of Q learning algorithms, they are not feasible for complicated problem statements with large number of states.
- Deep Q Networks show good performance in a large search spaces.
- Deep Q learning requires good amount of training before it can perform nicely enough.

VI. Future Work

- **LSTM:** Long Short Term Memory networks are known for performing good with sequential data. We feed our CNNs with 4 frames of the game which then gives a sense of direction and speed in the environment. We can use LSTM to avoid feeding 4 frames directly into a CNN and get better representation of those sequential frames.
- **DDQN:** Although the DQN algorithm is working remarkably well on the two environments, but it definitely has a scope of improvement due to its overestimation of values. The algorithm is known to learn unrealistically high action values at times as it involves a maximization step over all possible action values for a given state. To overcome this limitation of over-fitting, Double DQN algorithm [5] can be implemented.
- **RAM environments:** The environment we have currently used gives the game pixels as a state representation. These pixels are then needed to be fed to CNNs to get a consolidated state representation of the game. RAM environment provides you with a 128 byte representation of the Atari Machine RAM which can be considered as a consolidated version of game state and used for training the agent.

References

[1] Watkins, C.J.C.H. & Dayan, P. Mach Learn (1992) 8: 279. https://doi.org/10.1007/BF00992698

[2] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

[3] Clevert, D. A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289.

[4] R.S. Sutton and A. Barto. Reinforcement Learning: An Introduction. A Bradford Book. 1998

[5] H. van Hasselt, Arthur Guez and David Silver. Deep Reinforcement Learning with Double Q-Learning. AAAI, 2016.