UCLA Samueli Electrical & Computer Engineering

Double DQN for Overestimation Reduction in Games

Akhil Avula, Calvin Chang, Daniel Truong

ECE 239AS Reinforcement Learning, MS in Signals and Systems

Our group is in the process of re-implementing the proposed methods in van Hasselt et al.[2]to investigate how DDQN and DQN play different roles in overestimation. This will be achieved through simulation in an Atari game setting from gym.openai Brockman et al.[1]. In an attempt to improve the methods proposed in van Hasselt et al.[2], we propose two ideas to bring out new results: adjusting the hyper parameters or simulating in new environments. To evaluate the performance of the agents, DDQN and DQN will be pitted against each other, visualized through graphs and tables.

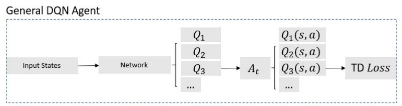
1. Introduction:

For this project, algorithms are adapted from"Deep Reinforcement Learning with Double Q-learning" from van Hasselt et al.[2]which proposes a solution called "Double DQN" to prevent overestimating values. The methods from van Hasselt et al.[2]report that Double DQN leads to better policies in video game environments than the DQN method. The DQN and Double DQN algorithm have mostly been replicated and while introducing a variation in the algorithms presented in van Hasselt et al.[2]to further investigate the issue of overestimation.

Why does DQN overestimate?

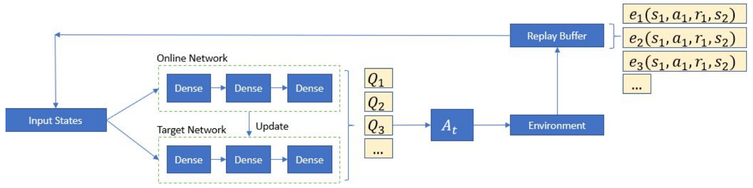
Overestimation is a substantial flaw in the broad spectrum of Q-Learning methods. These overestimated results are from a positive bias that is introduced because Q-learning uses the maximum action value as an approximation for the maximum expected action value.

Overestimation combined with bootstrapping propagates the wrong relative information about which states are more valuable than others, degrading the quality of the learned policies



2. Methods

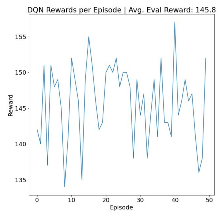
To improve overestimation double DQN was implemented. Pytorch was used to implement a three layer network double DQN. Below is a block diagram of our model. The network is interchangeable depending on the observation space. In this cases it is composed of linear layers. For environments with observations with images, the network was composed of convolutional layers.

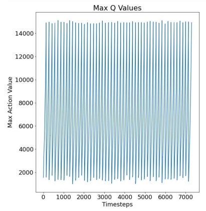


3. Results

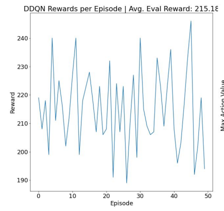
To compare the two algorithms, Opengym AI's cartpole environment was used to evaluate the performance of the two agents after using the same hyperparameters.

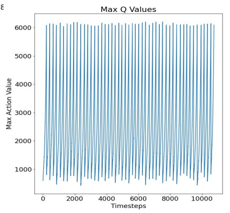
DQN Cartpole Evaluation Results



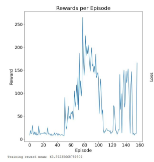


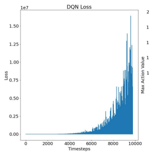
DDQN Cartpole Evaluation Results

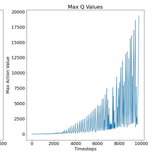




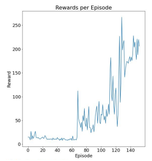
DQN Cartpole Training Results

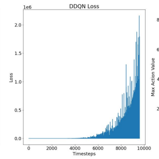


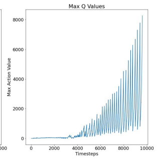




DDQN Cartpole Training Results







4. Discussion and Next steps

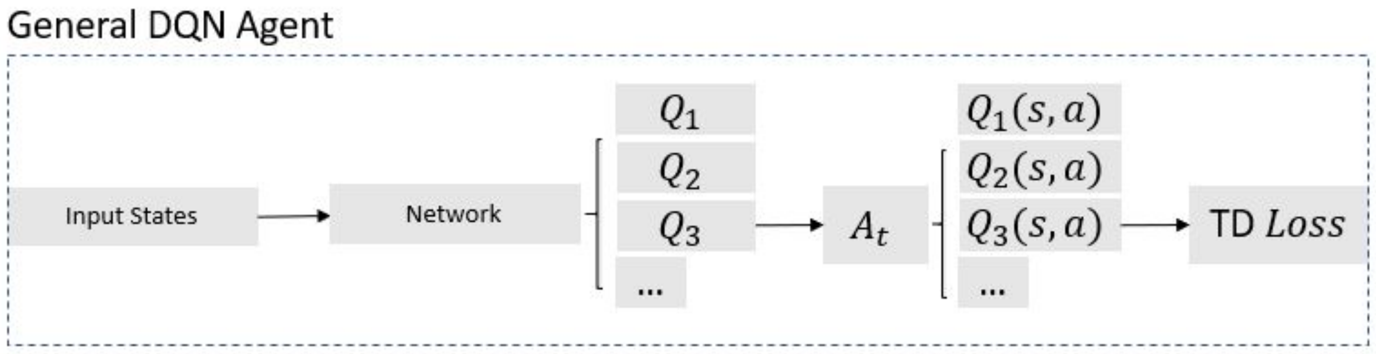
The DDQN not only performs better in the evaluation stage, but appears to be more stable in the training phase than the DQN. Comparing the plots of the max Q values for each algorithm during the evaluation stage, DDQN hovers from 1000-6000 while DQN hovers between 2000 to 14000. The overestimation of the DQN is shown here, which leads to worse performance than the DDQN. Another interesting result is that during the training phase, the overestimation of reward of DQN was followed by a strong dip in reward which can be linked the inaccurate Q value estimations.

This project can be taken further by training and evaluating both algorithms on more environments which have a similar learning difficulty to Cartpole, such as Pong, Lunar Lander, etc. Also, there are plans to tune the hyperparameters of the DDQN to show the potential benefit over DQN.

Why does DQN overestimate?

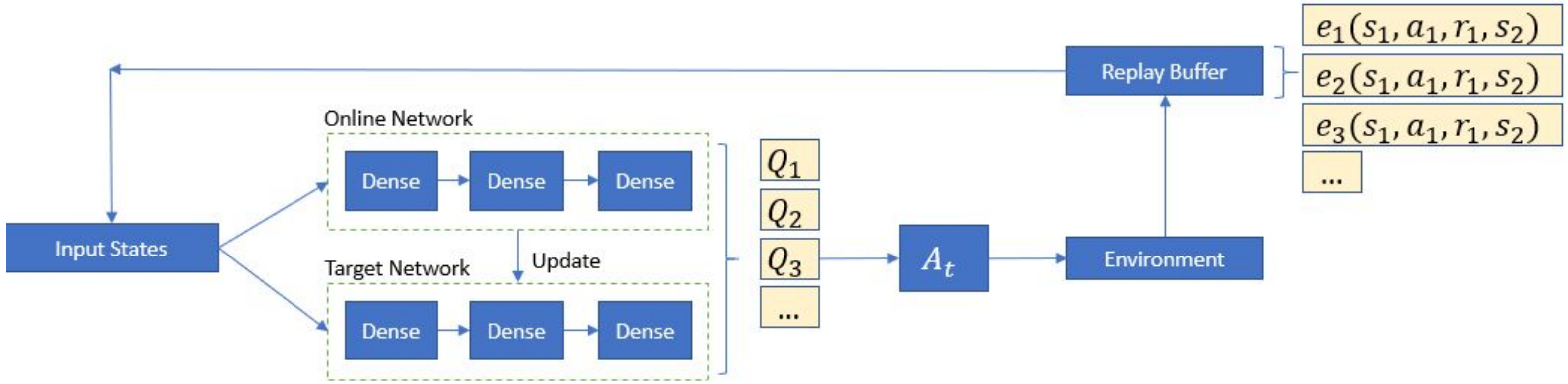
Overestimation is a substantial flaw in the broad spectrum of Q-Learning methods. These overestimated results are from a positive bias that is introduced because Q-learning uses the maximum action value as an approximation for the maximum expected action value.

Overestimation combined with bootstrapping propagates the wrong relative information about which states are more valuable than others, degrading the quality of the learned policies



2. Methods

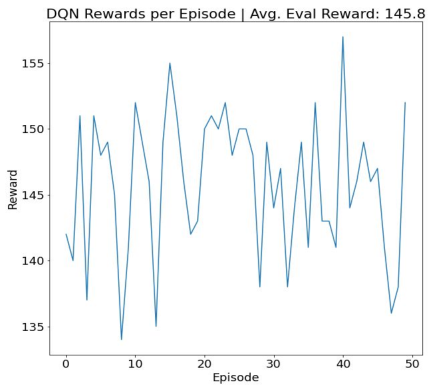
To improve overestimation double DQN was implemented. Pytorch was used to implement a double DQN. Below is a block diagram of our model. The network is composed of three layers that are interchangeable depending on the observation space. In this cases, it is composed of linear layers. For environments with observations that are images, the network was composed of convolutional layers.

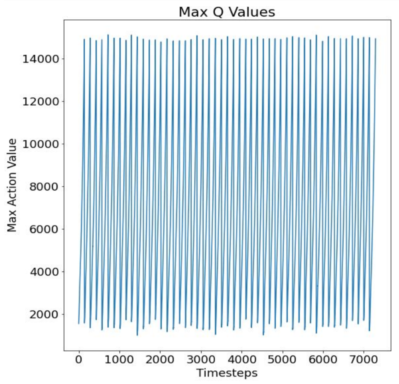


3. Results

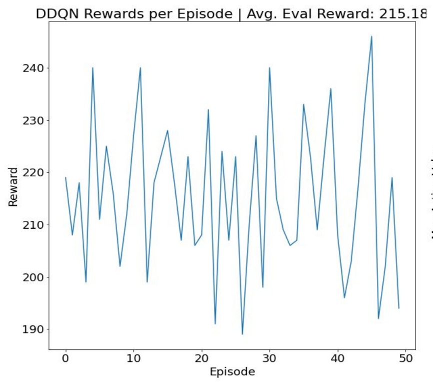
To compare the two algorithms, Opengym AI's cartpole environment was used to evaluate the performance of the two agents after using the same hyperparameters.

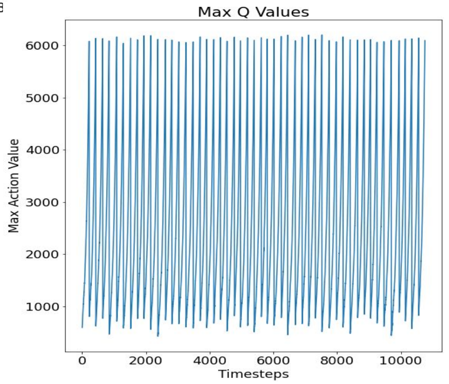
DQN Cartpole Evaluation Results



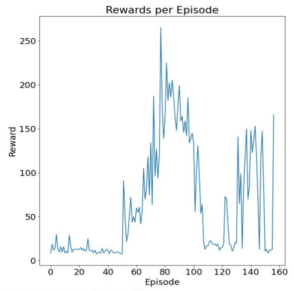


DDQN Cartpole Evaluation Results

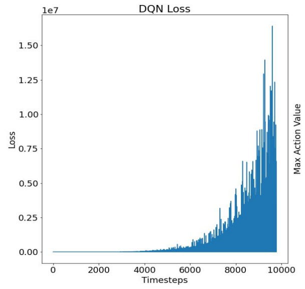


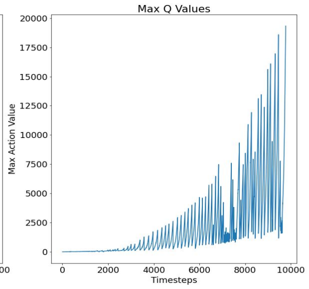


DQN Cartpole Training Results

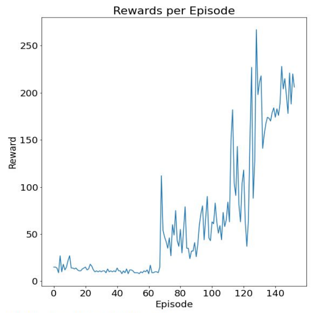


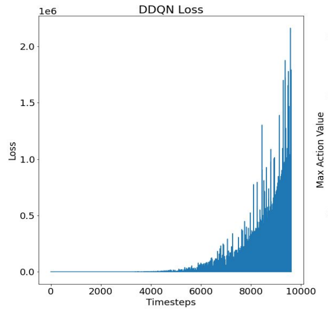
```
Training reward mean: 63.59235668789809
```

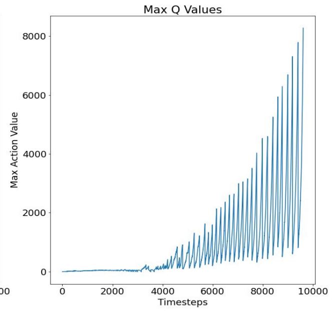




DDQN Cartpole Training Results







UCLA Samueli
Electrical & Computer Engineering

Double DQN for Overestimation Reduction in Games

Akhil Avula, Calvin Chang, Daniel Truong
ECE 239AS Reinforcement Learning, MS in Signals and Systems

Our group is in the process of re-implementing the proposed methods in van Hasselt et al.[2]to investigate how DDQN and DQN play different roles in overestimation. This will be achieved through simulation in an Atari game setting from gym.openai Brockman et al.[1]. In an attempt to improve the methods proposed in van Hasselt et al.[2], we propose two ideas to bring out new results: adjusting the hyper parameters or simulating in new environments. To evaluate the performance of the agents, DDQN and DQN will be pitted against each other, visualized through graphs and tables.

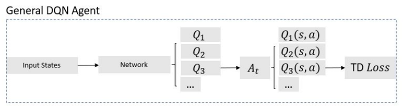
1. Introduction:

For this project, algorithms are adapted from"Deep Reinforcement Learning with Double Q-learning" from van Hasselt et al.[2]which proposes a solution called "Double DQN" to prevent overestimating values. The methods from van Hasselt et al.[2]report that Double DQN leads to better policies in video game environments than the DQN method. The DQN and Double DQN algorithm have mostly been replicated and while introducing a variation in the algorithms presented in van Hasselt et al.[2]to further investigate the issue of overestimation.

Why does DQN overestimate?

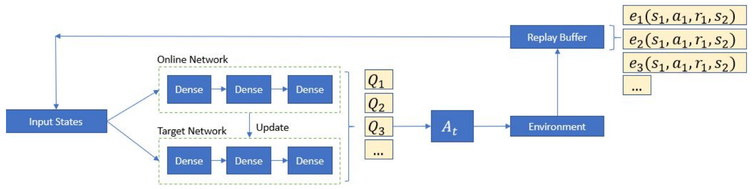
Overestimation is a substantial flaw in the broad spectrum of Q-Learning methods. These overestimated results are from a positive bias that is introduced because Q-learning uses the maximum action value as an approximation for the maximum expected action value.

Overestimation combined with bootstrapping propagates the wrong relative information about which states are more valuable than others, degrading the quality of the learned policies



2. Methods

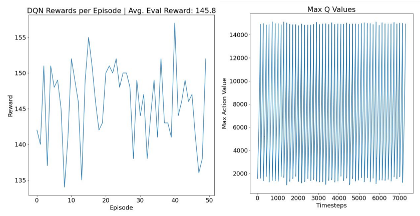
To improve overestimation double DQN was implemented. Pytorch was used to implement a three layer network double DQN. Below is a block diagram of our model. The network is interchangeable depending on the observation space. In this cases it is composed of linear layers.



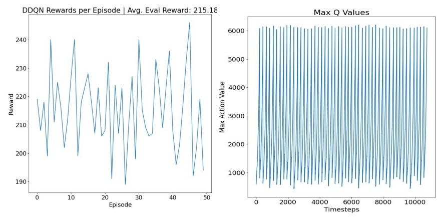
3. Results

To compare the two algorithms, Opengym AI's cartpole environment was used to evaluate the performance of the two agents after using the same hyperparameters.

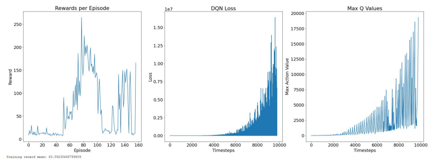
DQN Cartpole Evaluation Results



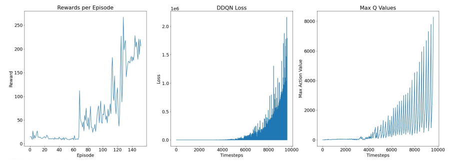
DDQN Cartpole Evaluation Results



DQN Cartpole Training Results



DDQN Cartpole Training Results



4. Discussion and Next steps

The DDQN not only performs better in the evaluation stage, but appears to be more stable in the training phase than the DQN. Comparing the plots of the max Q values for each algorithm during the evaluation stage, DDQN hovers from 1000-6000 while DQN hovers between 2000 to 14000. The overestimation of the DQN is shown here, which leads to worse performance than the DDQN. Another interesting result is that during the training phase, the overestimation of reward of DQN was followed by a strong dip in reward which can be linked the inaccurate Q value estimations.

This project can be taken further by training and evaluating both algorithms on more environments which have a similar learning difficulty to Cartpole, such as Pong, Lunar Lander, etc. Also, there are plans to tune the hyperparameters of the DDQN to show the potential benefit over DQN.