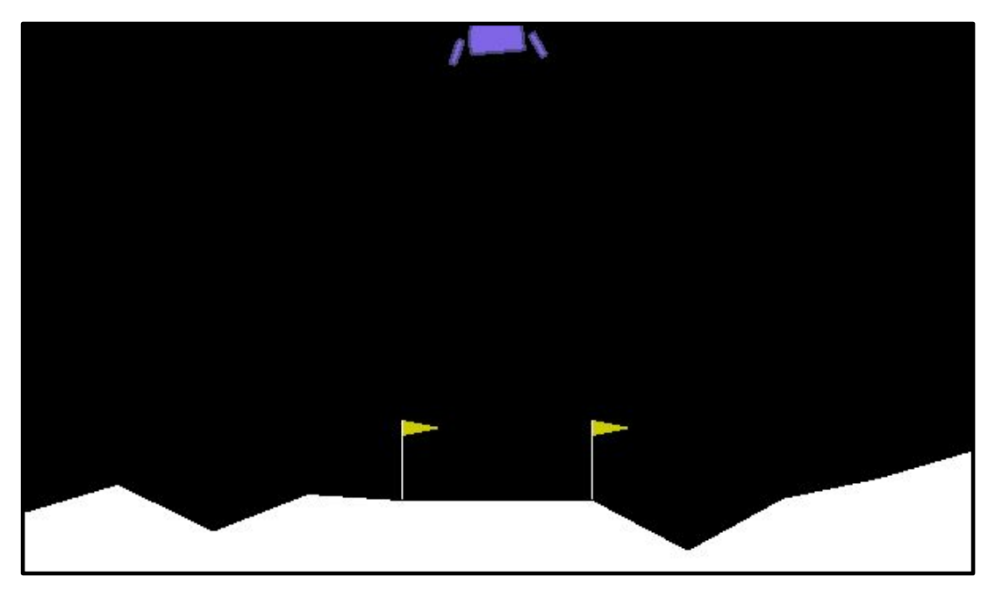
A Moonshot to the Moon

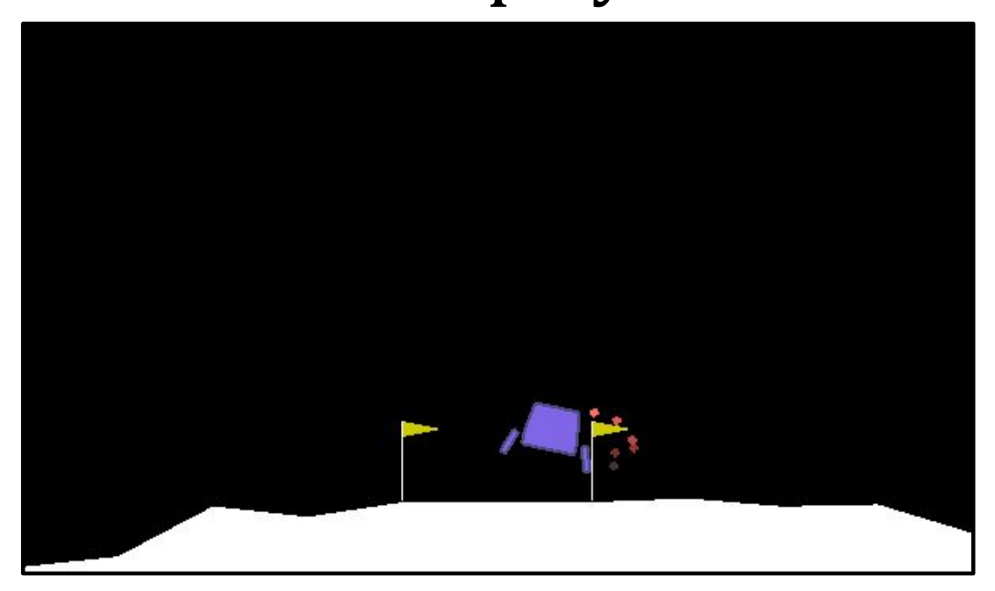
Carlos Gonzalez, Dan Burke, Rebecca Lee

Lunar Lander v2

Initial State



Gameplay

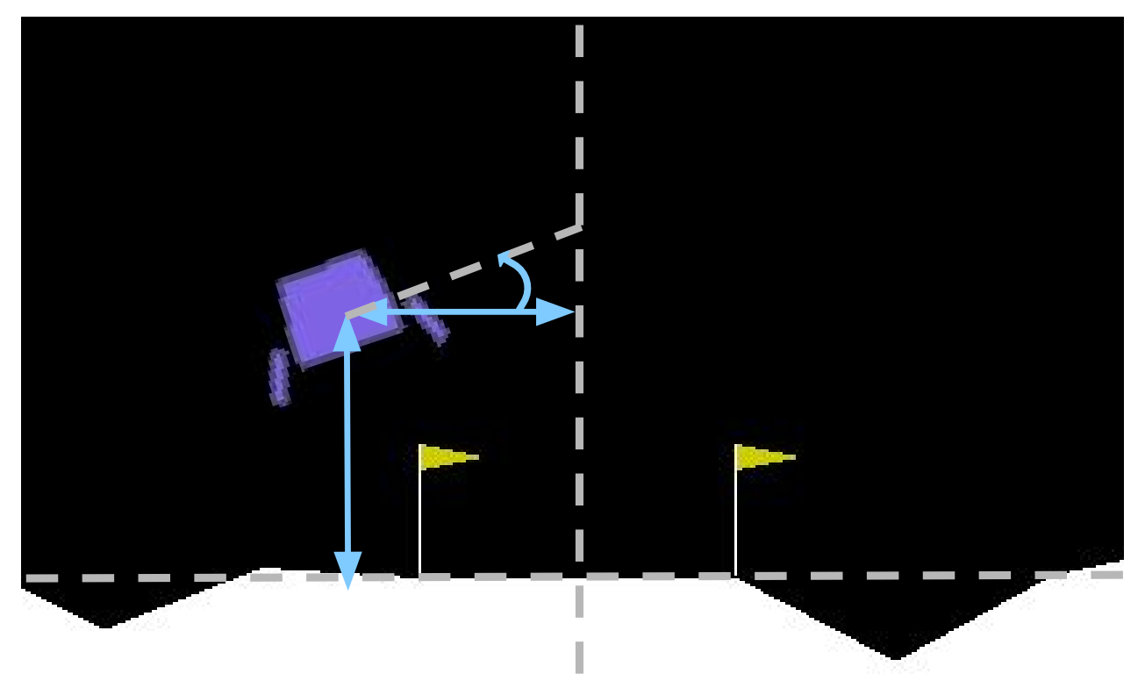


The goal of Lunar Lander is to land in the flagged-marked region with low velocity and normal orientation. The task requires the agent to learn the optimal policy in the simple physics based environment. We decided to approach this problem with Q-learning, a reinforcement learning algorithm.

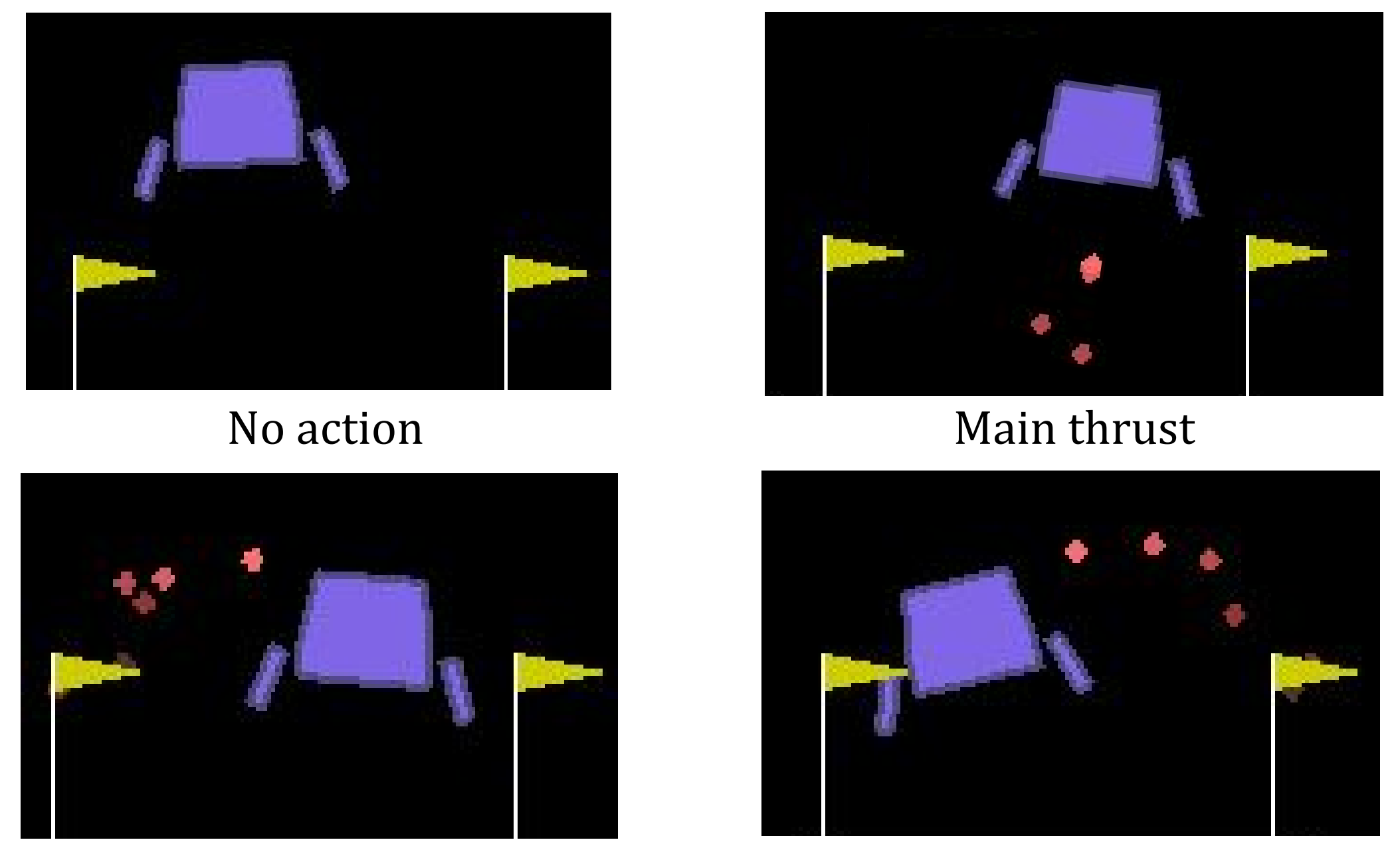
State Space & Action Space

State Vector

0 - Horizontal Distance (x)
1 - Vertical Distance (y)
2 - Horizontal velocity
3 - Vertical velocity
4 - Angle (θ)
5 - Angular velocity
6 - Right leg touching ground
7 - Left leg touching ground



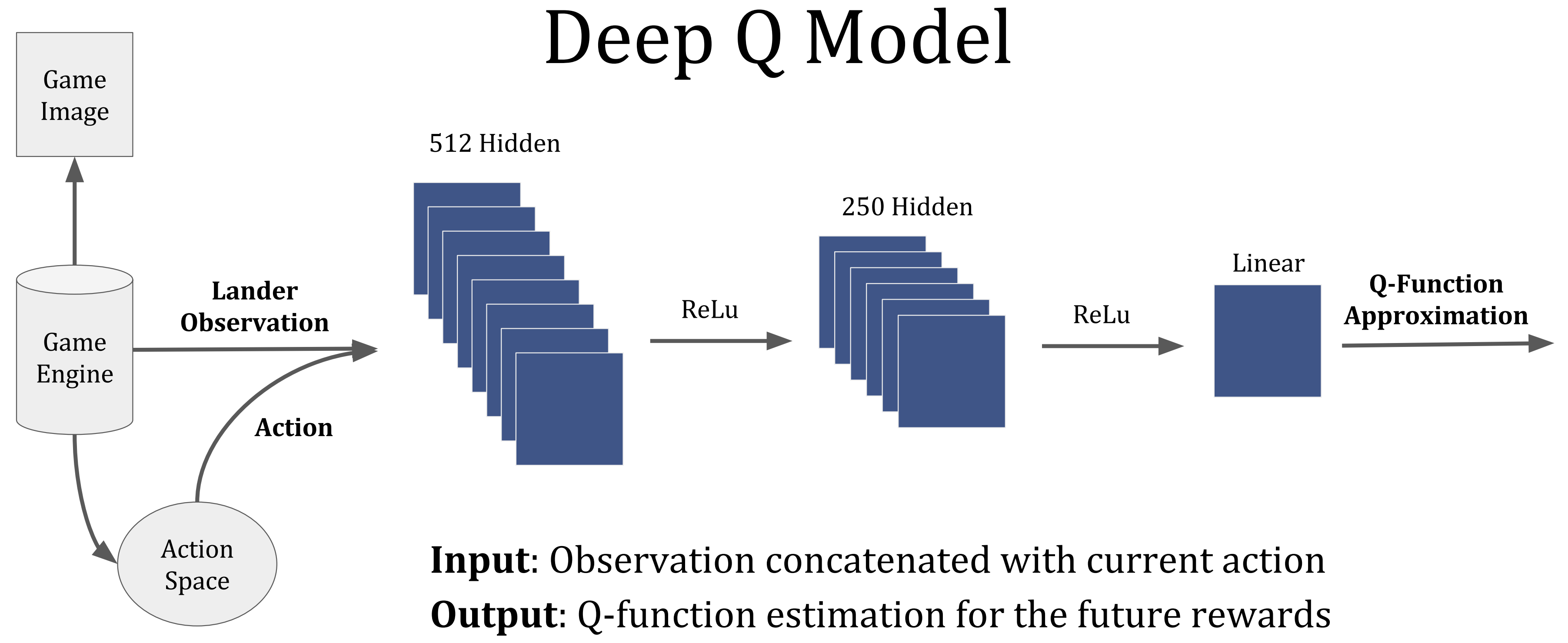
Actions



Approach Summary

- To explore the field, we started by implementing our own hand crafted Deep Q-Learning models
- We experimented with learning Q functions and also a more advanced version of learning Q values from states directly.
- Finally, we explored advanced Deep RL models

Deep Q Model

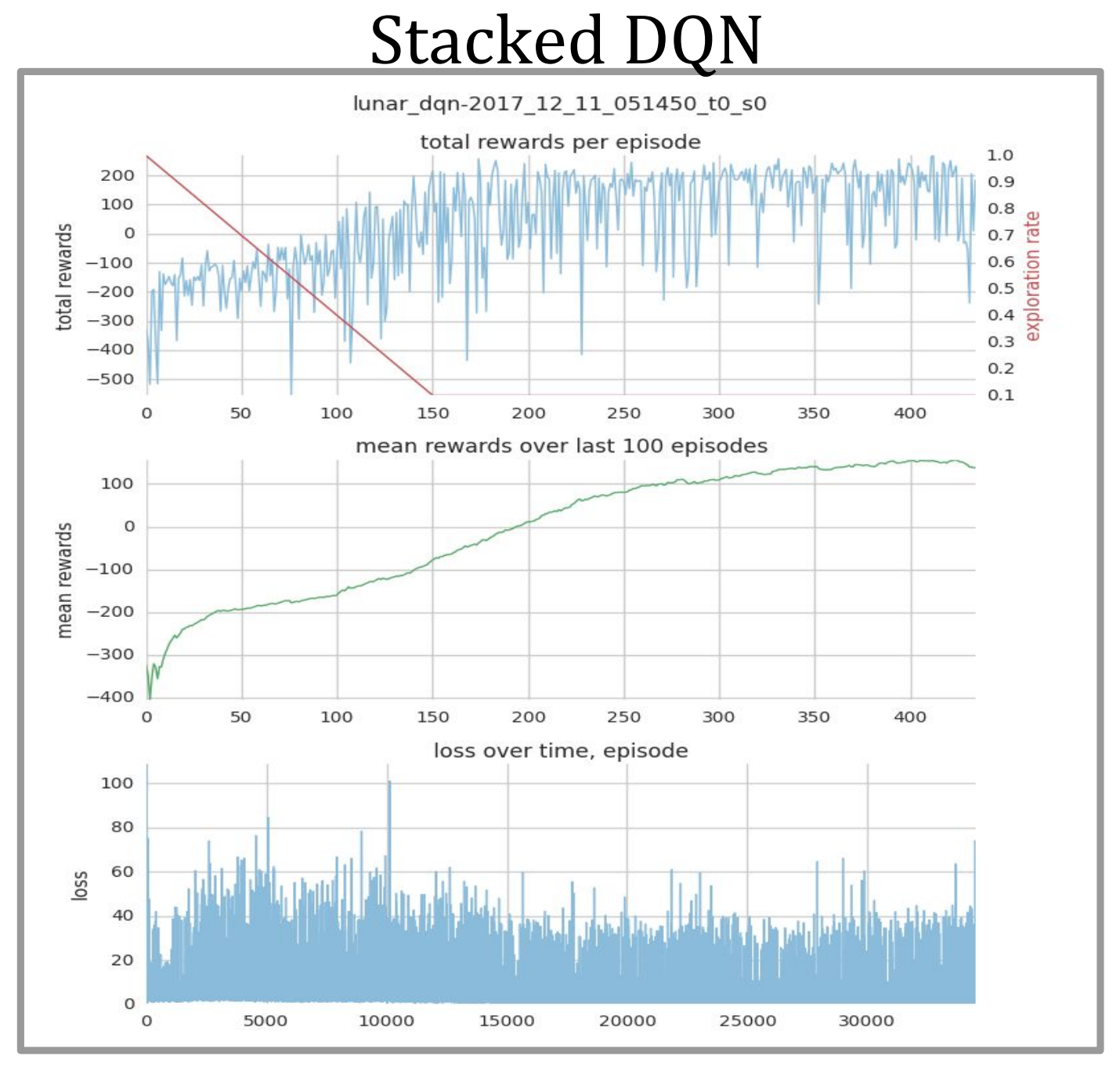


Input: Observation concatenated with current action
Output: Q-function estimation for the future rewards

- Sample generation: Training sample "ground truth" created by a manual walkthrough of Bellman's after an episode of the game
- Training: Samples are fetched from experience replay storage and model is retrained every 10 episodes

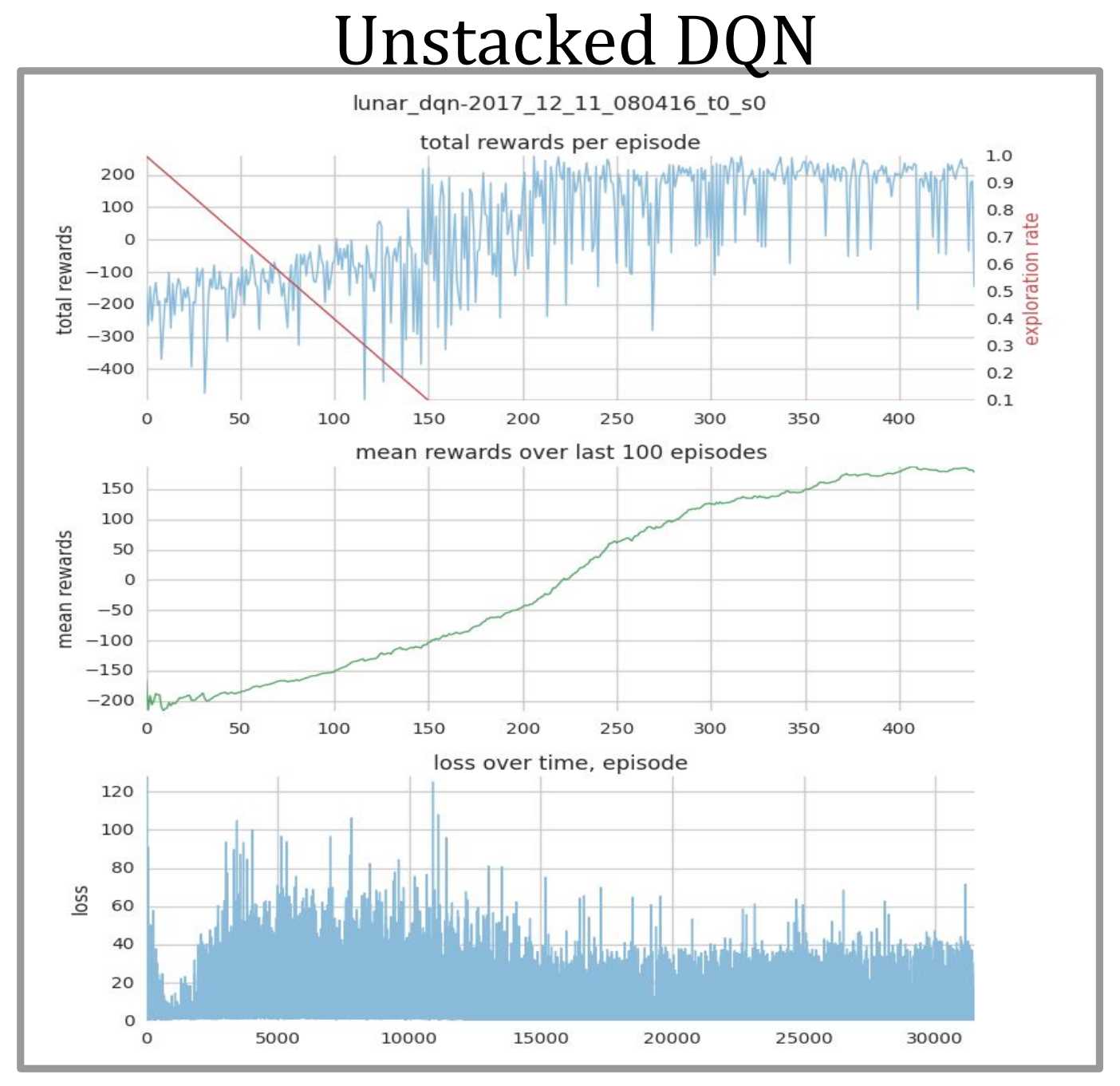
Advanced Models

Stacked DQN



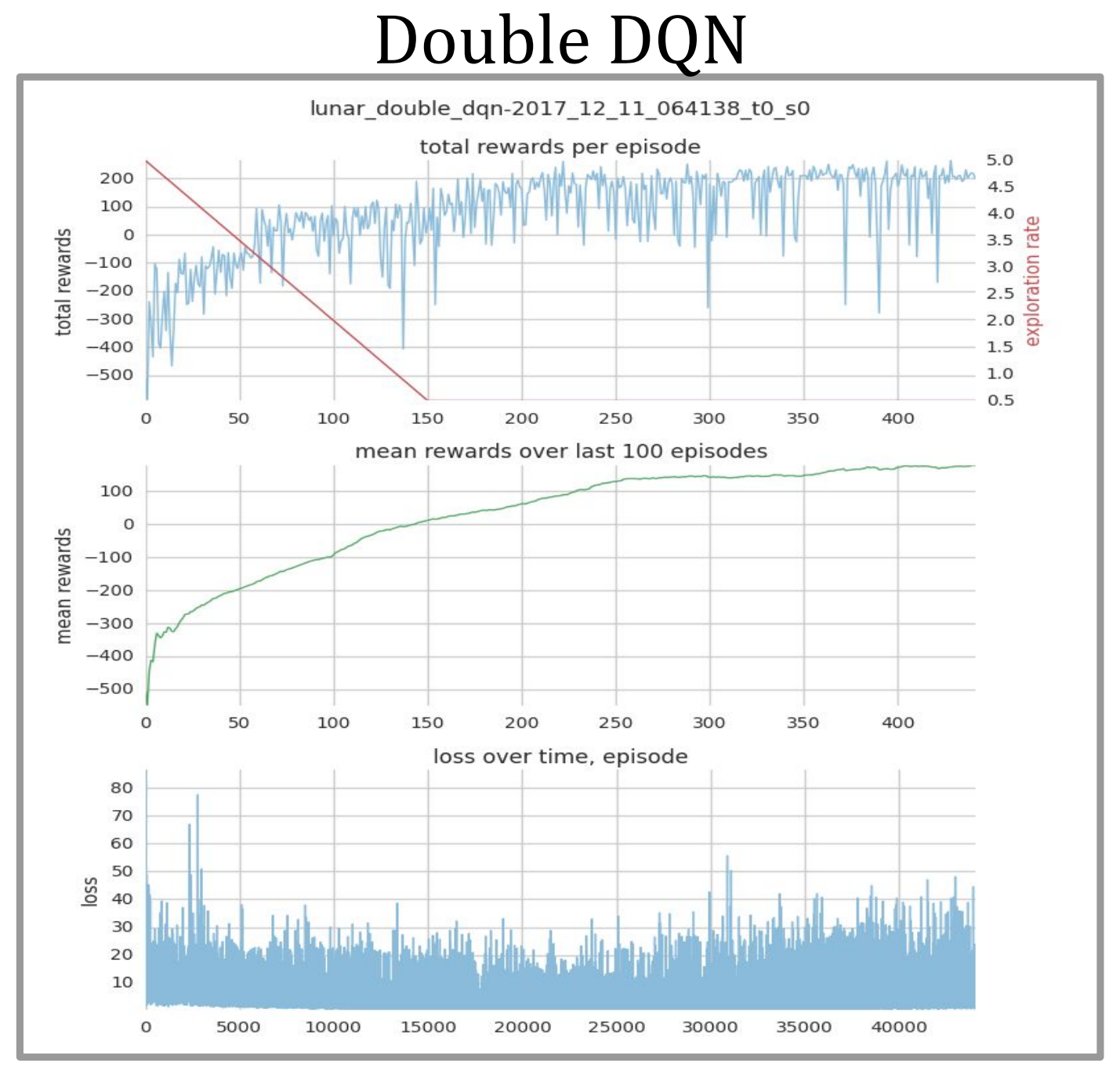
- Adam Optimizer, Epsilon Greedy Policy
- Hidden layers: [400,200]

Unstacked DQN



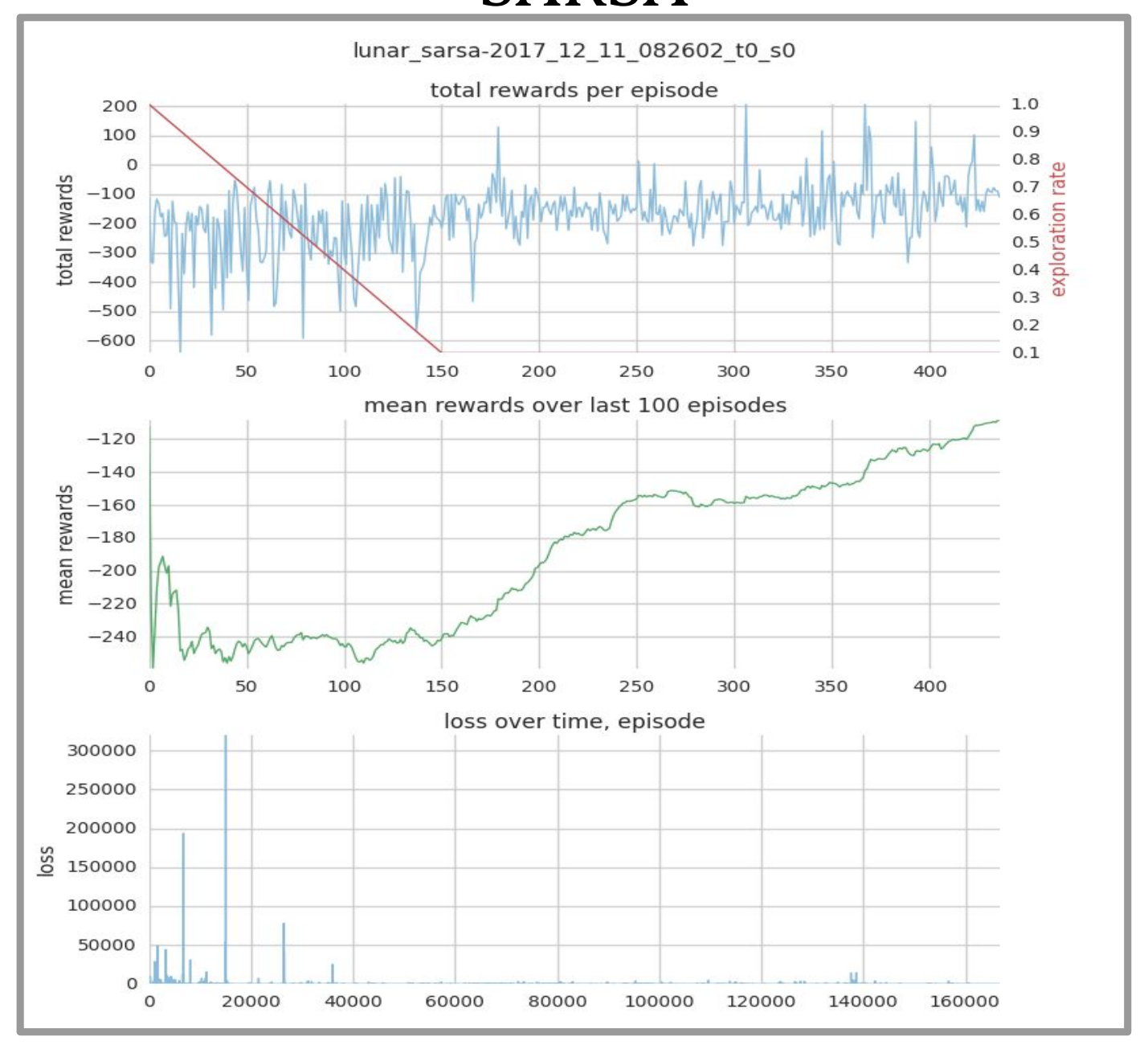
- Adam Optimizer, Epsilon Greedy Policy
- Hidden layers: [400,200]

Double DQN



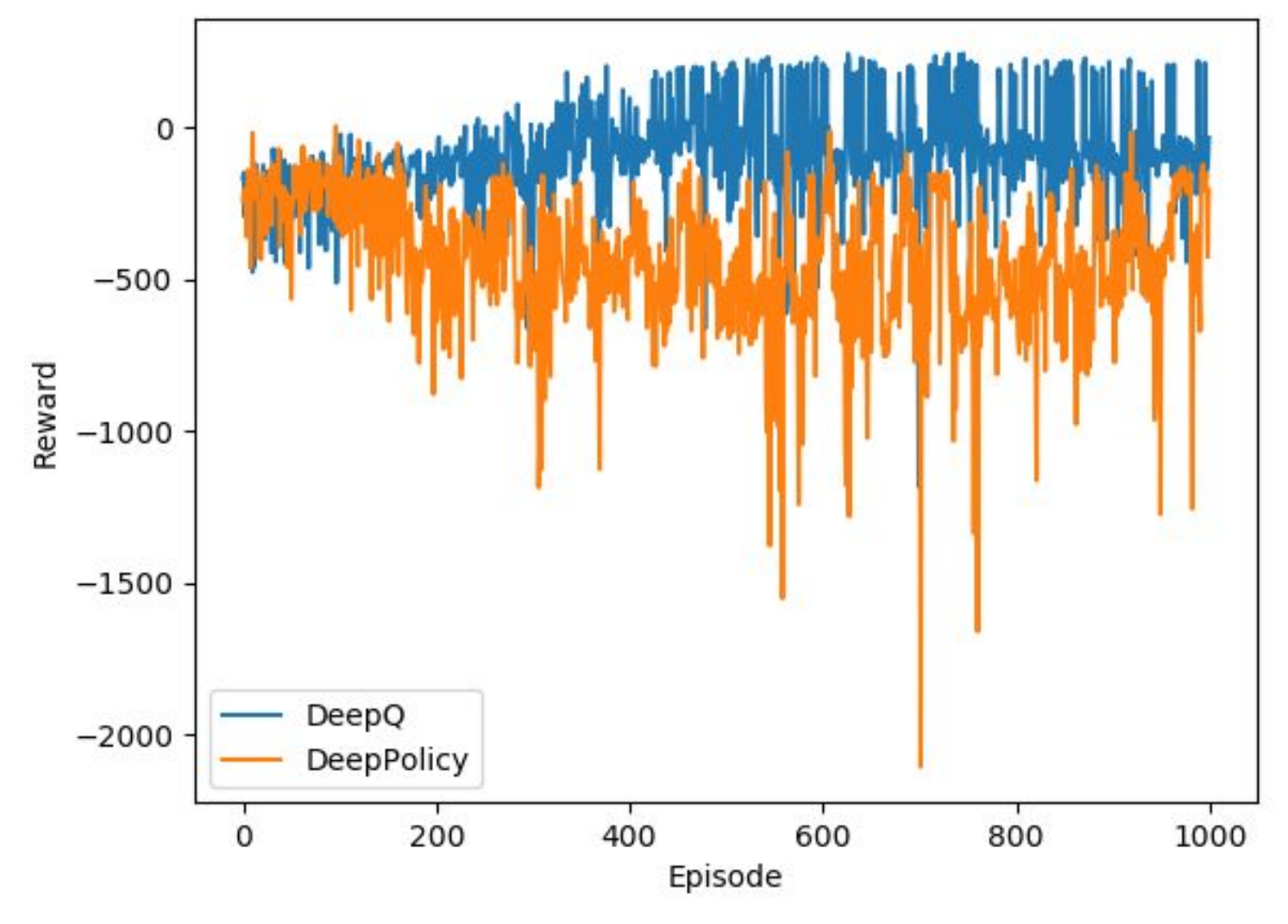
- Adam Optimizer, Double DQN Boltzmann Policy
- Hidden layers: [800,400]

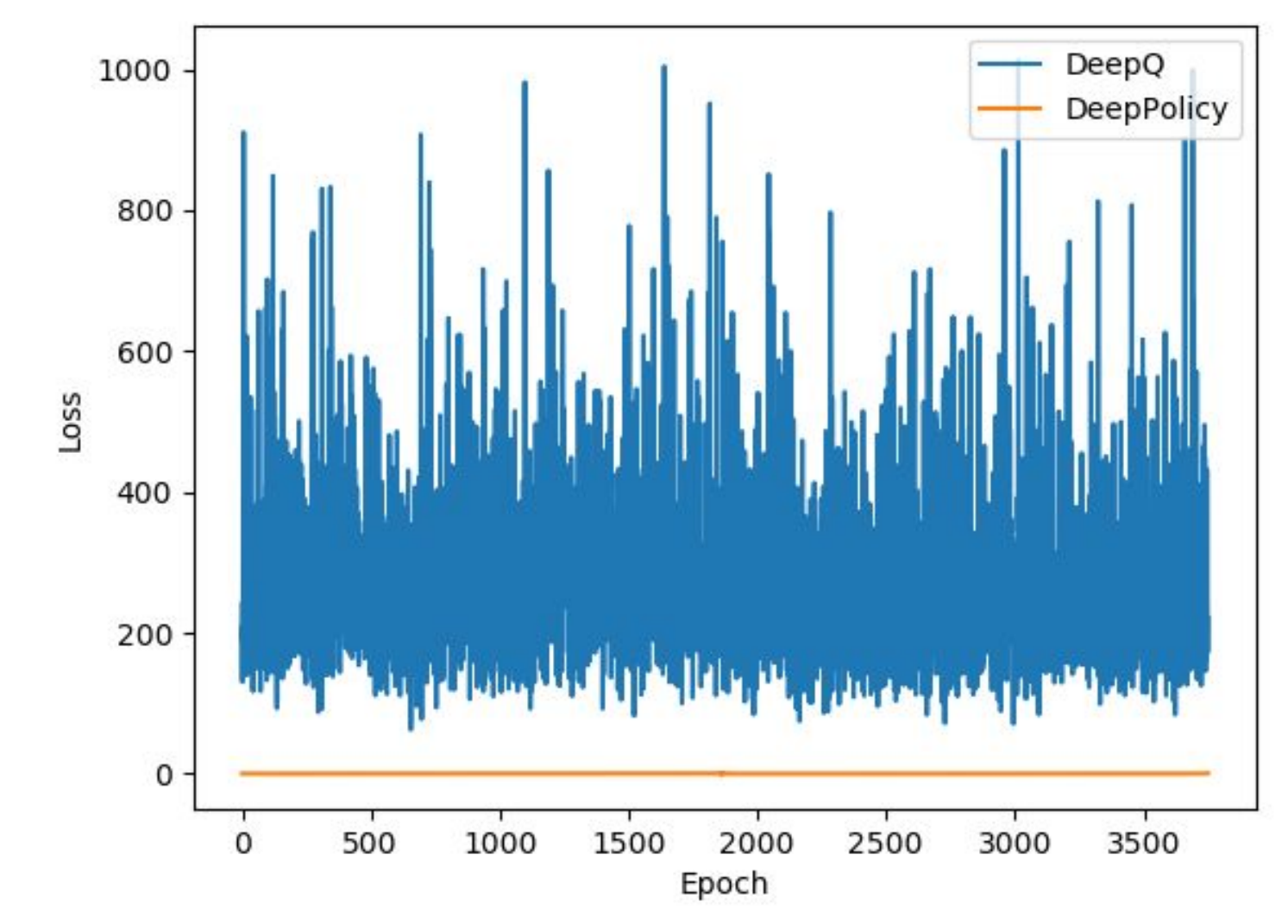
SARSA



- Adam Optimizer, Epsilon Greedy Policy
- Hidden layers: [300, 150, 75]

Deep Q Function Comparisons





- Learning the Q function with a state and action as a parameter is clearly much stronger with chosen parameters
- Learning all actions simultaneously seems more general and likely converges to a better result later on
- Smaller gradients lead to slower learning

Challenges & What We Learned

- Convergence is a big issue
- Reaching local maximas
- Reward functions behaved as bandaids for weaker models
- Experience replay is a must
  - Could learn basic tasks better
  - Still struggled with complexity
- No difference between using change in rewards vs. current rewards
  - Temporal observation features take this into account

Acknowledgements

We utilized OpenAI Gym for the RL toolkits and its provided Lunar Lander environment. OpenAI Lab provided an easy interface to OpenAI gym with advanced models and automated experimentation.

OpenAI Lab

OpenAI