Training a Deep Reinforcement Learning Agent for Doom

Author: Daniel Netzl

Student ID: K17705867

1. Introduction

This project aimed to apply DRL techniques to train an agent for the game Doom. The goal was to implement an agent that could learn effective combat policies against automated bots.

2. Methodology & Training Journey

The core of my approach was a Dueling Deep Q-Network (Dueling DQN). This architecture was chosen for its ability to efficiently learn state values and action advantages separately, which can lead to better policy evaluation in action-rich environments. The network processed visual input from the game, specifically 128x128 pixel frames, through a series of convolutional layers to extract relevant spatial features. These features were then fed into two separate fully connected streams: one estimating the state value (V(s)) and another estimating the advantage for each action (A(s,a)). These were combined to produce the final Q-value estimates.

The agent operated within the VizdoomMPEnv environment, specifically on the "ROOM" map populated with four enemy bots. I would have liked to provide the agent with temporal information for understanding motion and dynamics, but frame stacking led to several errors, which made me stick with only 1 frame as input to the network. Instead of RGB I opted for the grayscale screen buffer for training speed. A depth buffer was utilized to give the agent explicit information about distances to objects and surfaces in its vicinity.

A crucial aspect of the agent's development was the design of the reward function. I designed a multi-component reward signal that included: a significant positive reward for each enemy frag (+100), a smaller positive reward for each successful hit on an enemy (+5), a minor penalty for taking damage (-0.1), a minor penalty for being alive (-0.005), and a positive bonus for movement. The movement bonus was only counted if the movement was followed by a hit or frag. The bonus doubled the hit reward and added +20 to a subsequent frag. This way, the agent should be encouraged to move and discouraged to just stand and shoot and hope for lucky hits. The latter behavior was observed during a previous training phase (without movement bonus and longevity penalty), making the agent stuck in local optima where lucky hits were more frequent than expected.

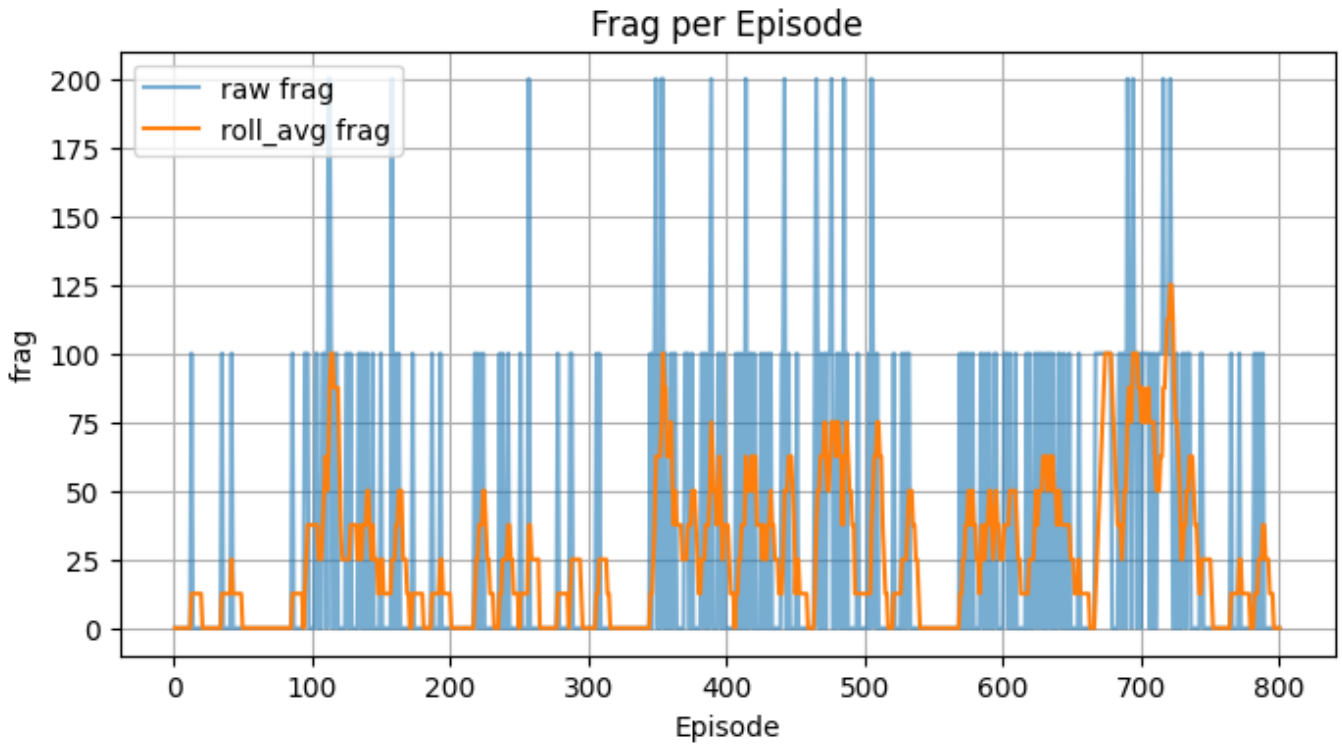
The training process utilized an experience replay buffer to store past transitions (state, action, reward, next state, done). During each learning step, mini-batches of these transitions were sampled to update the Dueling DQN's parameters via the Adam optimizer. To stabilize learning, a target network, which was a periodically updated copy of the main network (using Exponential Moving Average - EMA), provided consistent targets for the Q-value updates. Exploration was managed using an epsilon-greedy strategy, where the agent initially took random actions with high probability (epsilon = 1.0) and gradually reduced this randomness (decaying epsilon to 0.1) to exploit its learned knowledge. Key hyperparameters included a learning rate of 1e-4, a discount factor (gamma) of 0.999, and a batch size of 32. The buffer size was initially set to 10,000 and was subsequently increased to 20,000 in the latest training run. Epsilon decayed slowly by 0.99997 each step.

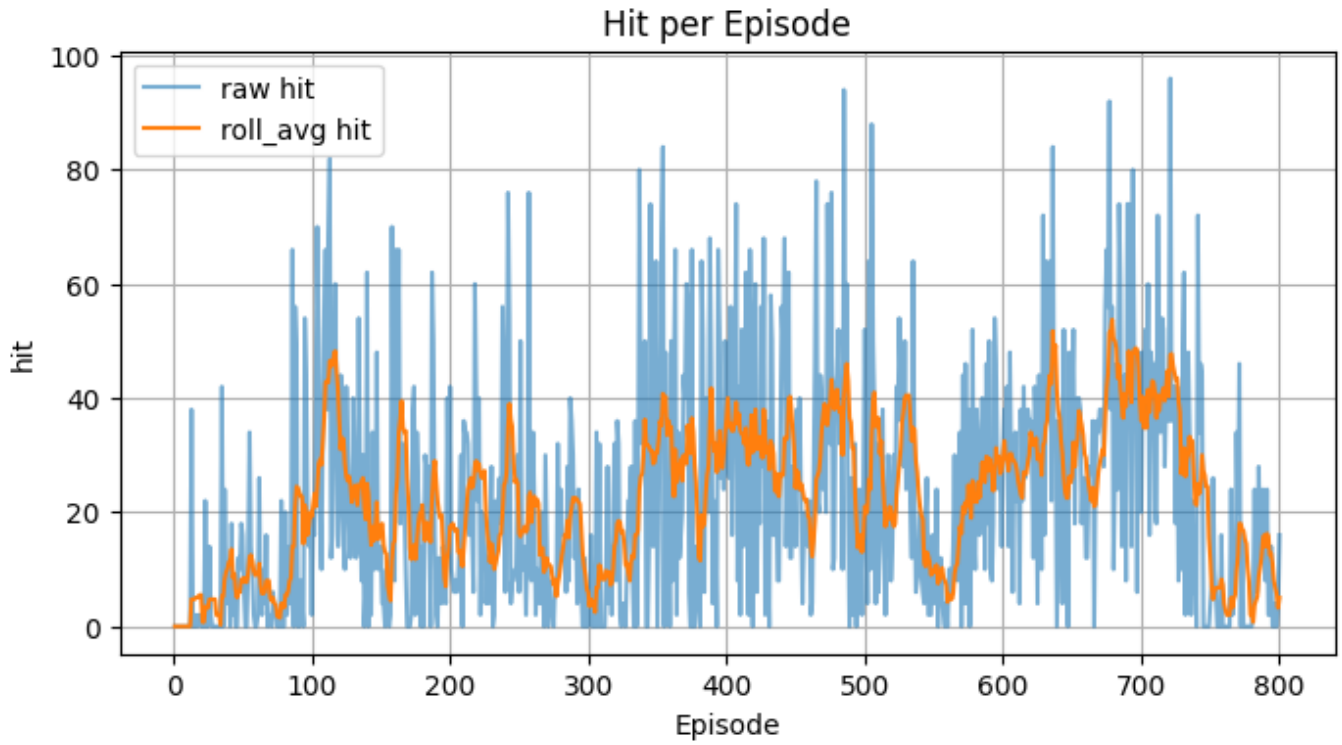
The first mayor run was done with roughly 2,000 episodes of training. The reward function, which at this point was still the basic one provided, did not provide enough incentives for the agent to learn to explore and understand its targets. Also, including PER and N-step rewards did not improve the resuts by too much. Just after the change of the reward function and incentivizing movement with a bonus, the agent's behavior drastically changed and improved. This way, the agent learned quickly, landing hits and frags relatively consistentnly a few hundred episodes in. And it only took 480 episodes for agent to score the max points in a 800 episdoes long training cycle.

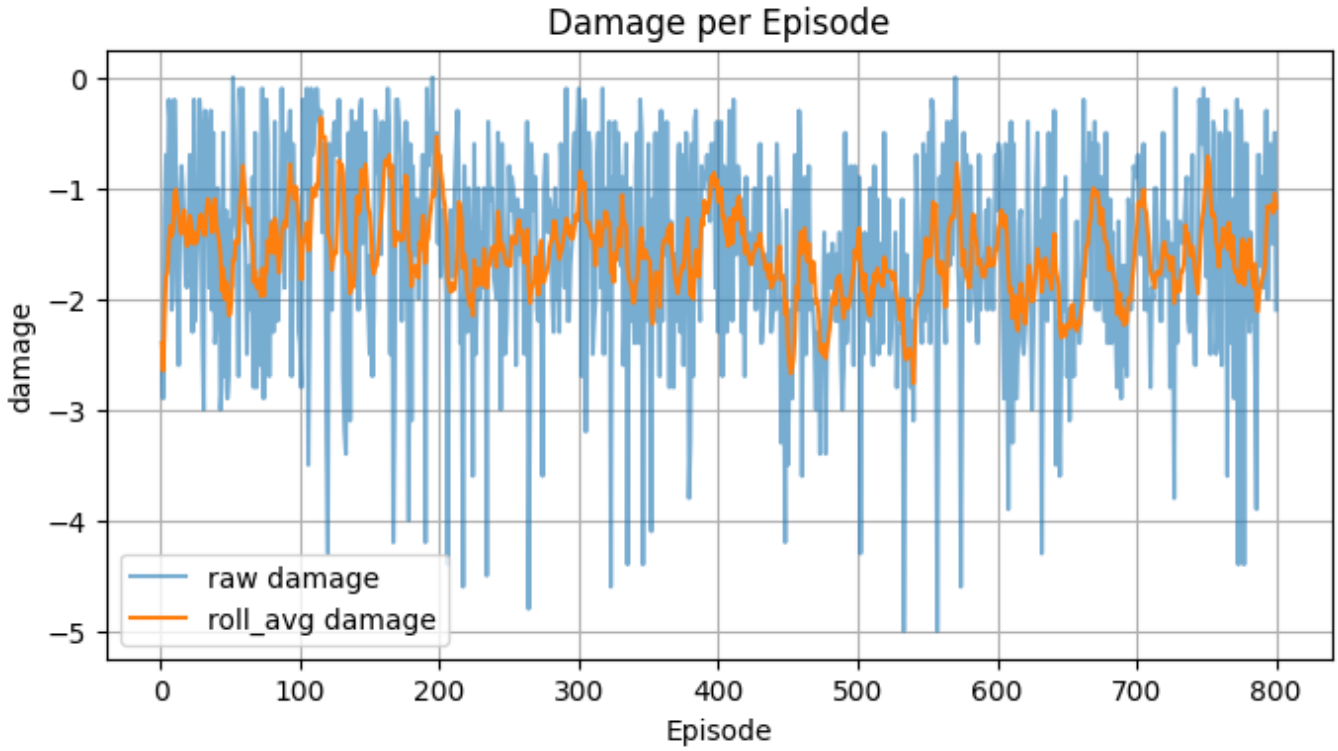
3. Results and Analysis

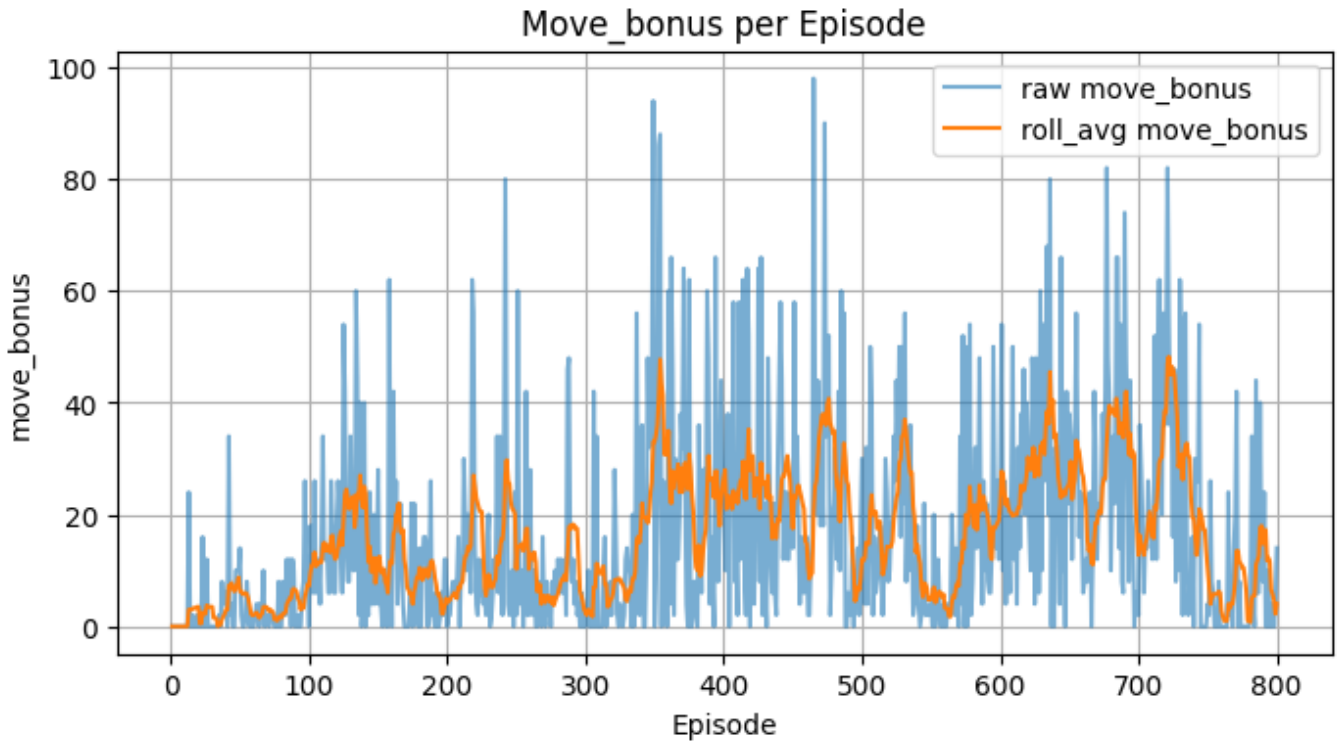
The iterative development process culminated in a highly effective agent that scored 75 points on the challenge server. Locally, the best performing model was identified at episode 480, achieving an evaluation shaped return of 308.80.

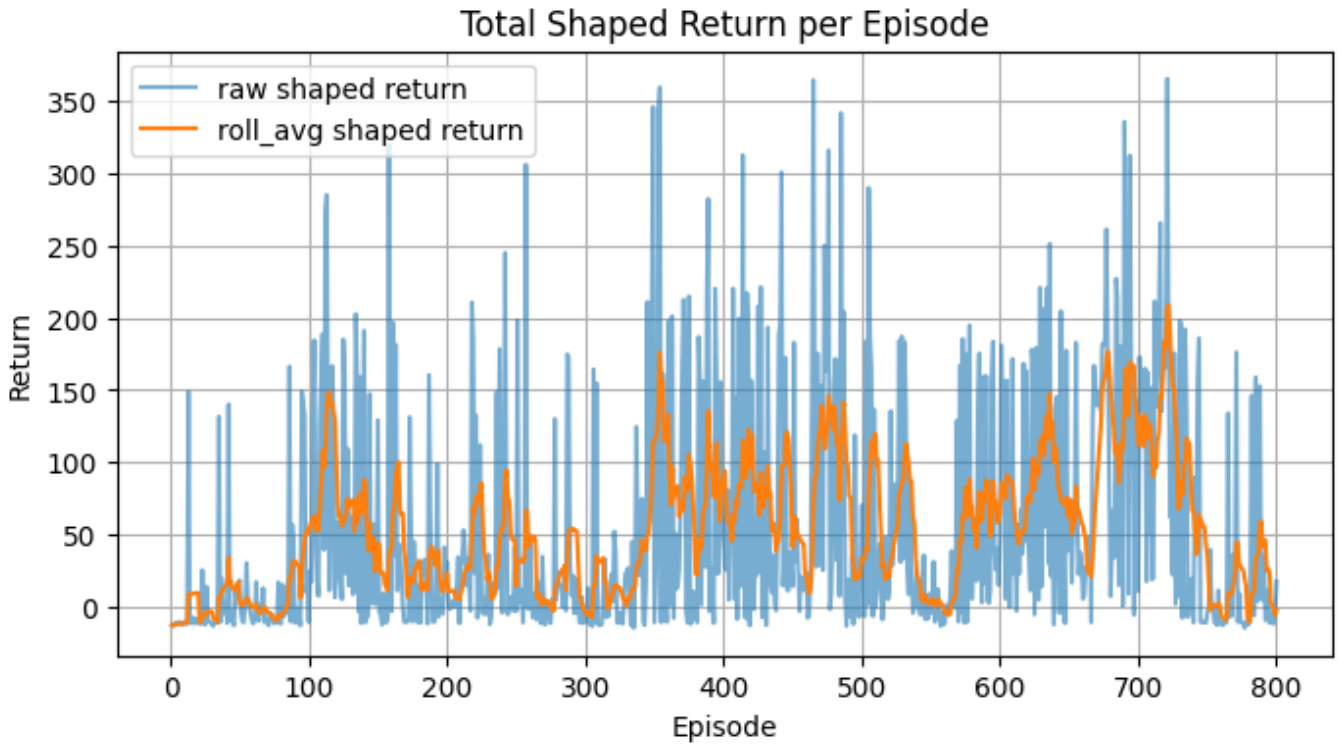
Analysis of the training metrics provides further insights into the agent's learning progression. The plots for "Frag per Episode" and "Hit per Episode" showed clear upward trends in their rolling averages, indicating that the agent progressively improved its ability to successfully engage and eliminate targets. The "move_bonus per Episode" plot also demonstrated a positive learning curve, suggesting the agent learned to incorporate more active movement into its strategy. Consequently, the "Total Shaped Return per Episode" showed a relatively consistent increase, with some phases of lower returns as well. Generally, the plots reflect the agent's growing proficiency in maximizing all beneficial reward components over time.


Frag per Episode
raw frag
roll_avg frag
frag
Episode
0
25
50
75
100
125
150
175
200
0
100
200
300
400
500
600
700
800


Hit per Episode
raw hit
roll_avg hit
hit
Episode
0
20
40
60
80
100
0
100
200
300
400
500
600
700
800

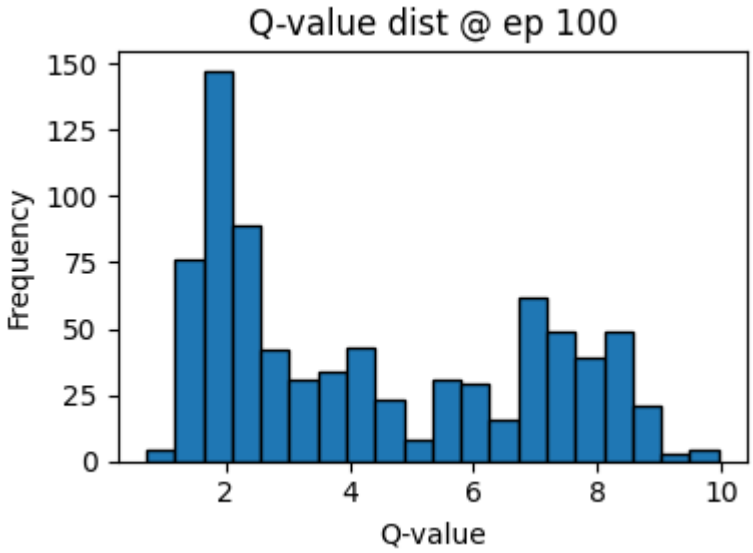

Damage per Episode
damage
Episode
raw damage
roll_avg damage
0
−1
−2
−3
−4
−5
0
100
200
300
400
500
600
700
800

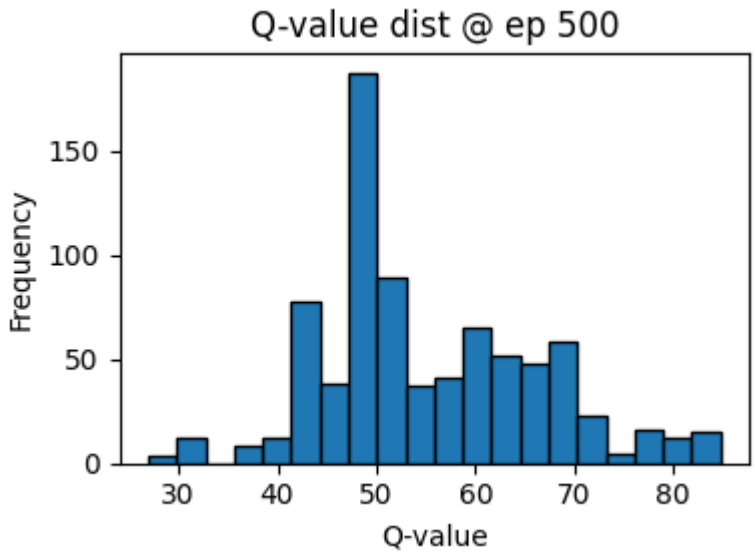

Move_bonus per Episode
move_bonus
Episode
raw move_bonus
roll_avg move_bonus
100
80
60
40
20
0
0
100
200
300
400
500
600
700
800

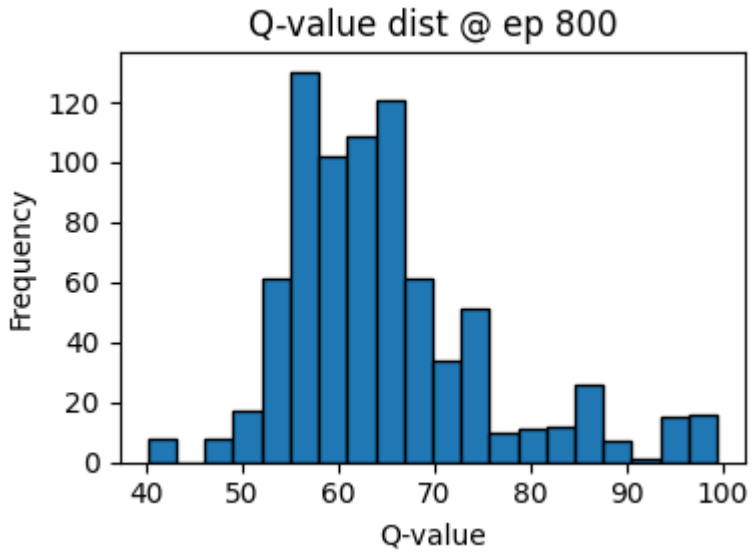


The Q-loss generally decreased after heavy initial fluctuations. The Q-value distribution plots, sampled every 100 episodes of training, showed a positive trend: the magnitude of predicted Q-values progressively increased. This is a strong indicator of learning, as it shows the agent associating game states with increasingly higher expected future rewards, mostly aligning with its improved performance in acquiring actual rewards. By episode 500, Q-values were significantly higher than in early training, reflecting the agent's enhanced understanding of the game's value landscape. However, towards the end of the 800-episode training cycle, the Q-values kept improving, while the agent's actual rewards partially decreased again. This could indicate that the agent started to overestimate the value of certain actions or states, or that the policy was subtly shifting towards less effective behaviors despite the network's optimistic predictions.


Q-Loss per Training Step
raw q-loss
roll_avg q-loss
Loss
Training Step
0
50
100
150
200
250
0
100000
200000
300000
400000


Q-value dist @ ep 100
Frequency
Q-value
0
25
50
75
100
125
150
2
4
6
8
10





The statistics from the best local model at episode 480 (total environment steps 1,008,000):

- epsilon: 0.1
- learning rate: 6.2e-05
- replay buffer size: 20,000
- Shaped Return: 153.90
- Avg Loss (approx): 15.3177
- Evaluation Shaped Return: 308.80