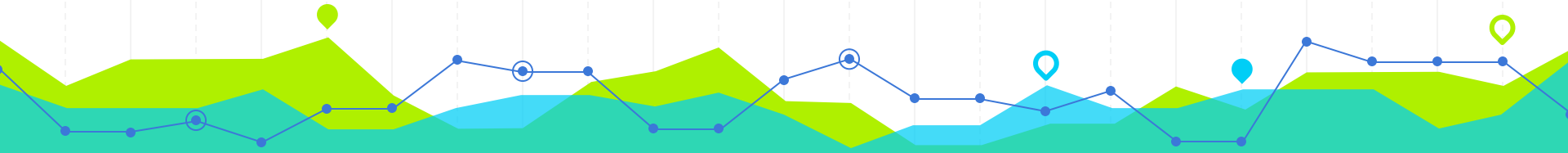
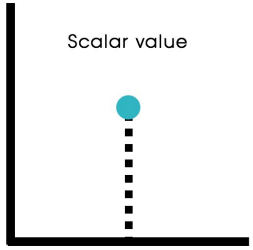
Granularity in Distributional RL



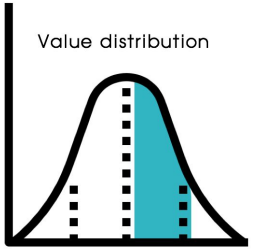
Rishi Advani
Ayush Gupta
Ahmed Sayeed Faruk

CS 594
04/27/2022

Distributional RL



General RL



Distributional RL

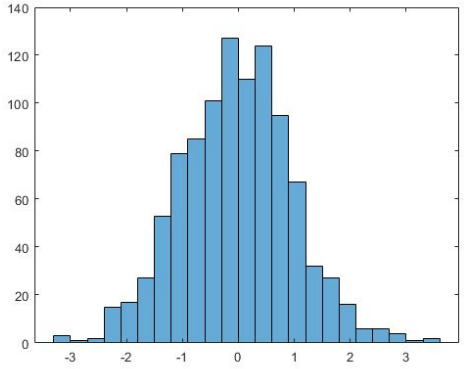
- What is distributional reinforcement learning?
  - Captures the entire reward distribution rather than just the mean.
  - E.g., standard Q-learning and Sarsa are not distributional.

- Benefits of distributional reinforcement learning
  - Preserving the distribution of rewards allows agents to have a robust performance.
  - Captures the randomness inherent to the rewards.

The Problem Statement

- For our project we sought to derive a relationship between the performance and
  1. the number of *bins* in the C51 algorithm.
  2. the number of *quantiles* in the QR-DQN algorithm.

Bins and Quantiles

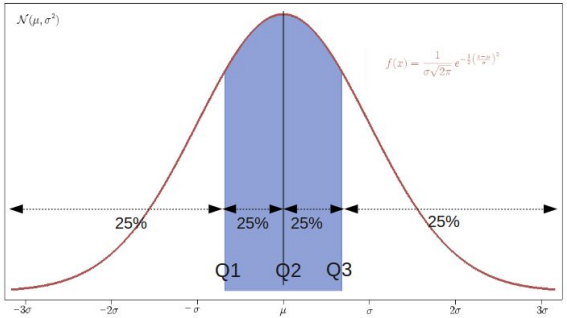


Bins

- Partition of data into equal-sized subsets

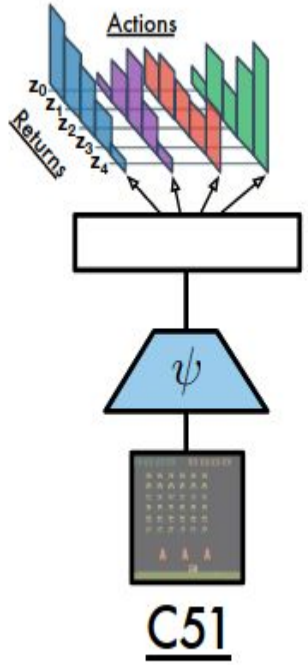
Quantiles

- Partition of data into subsets with equal probabilities



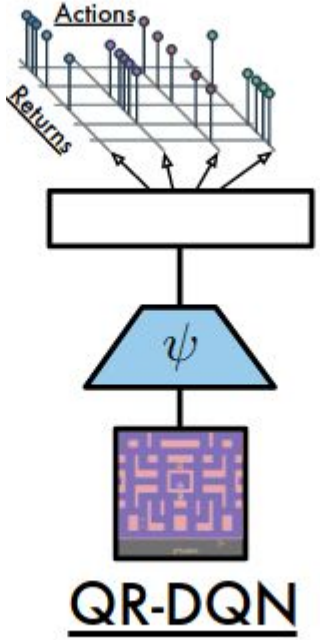
C51 Algorithm

- Q-learning-type algorithm based on DQN
- Computes the probability of the return falling into each bin, instead of computing the expected return
- KL divergence is used as the loss function
- The returns range from V_{min} to V_{max} which is split even by the number of atoms (N), in C51, N = 51



QR-DQN Algorithm

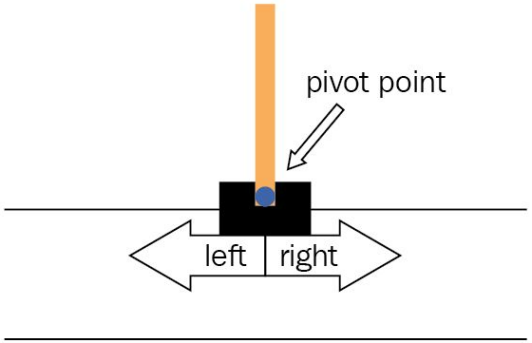
- Separates reward amounts into variable-sized bins with fixed probabilities (i.e., quantiles)
- Uses quantile regression to model the distribution over returns
- Uses a quantile Huber loss as the loss function
  - smoothed version of traditional quantile regression loss



Environment

CartPole-v1 Environment

- The goal is to keep the pole perpendicular to the cart to achieve max score
- A reward of +1 is provided for every timestep that the pole remains upright
- Action space: {Move left, Move Right}
- Maximum reward: 500



Environment Challenges

- Memory limits and bugs in dependencies of Atari environments in Google Colab

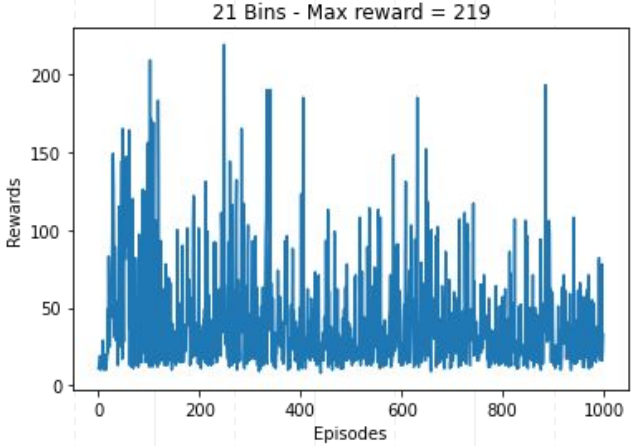
Results - C51

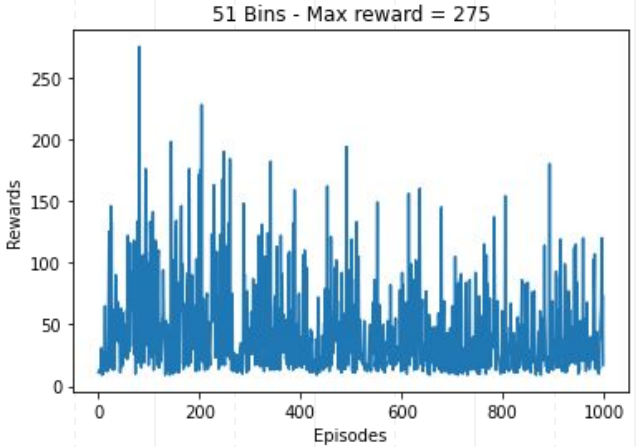
We are continuing our experiments, but at this point, we have not been able to detect any correlation between the number of bins and the performance of the agents.

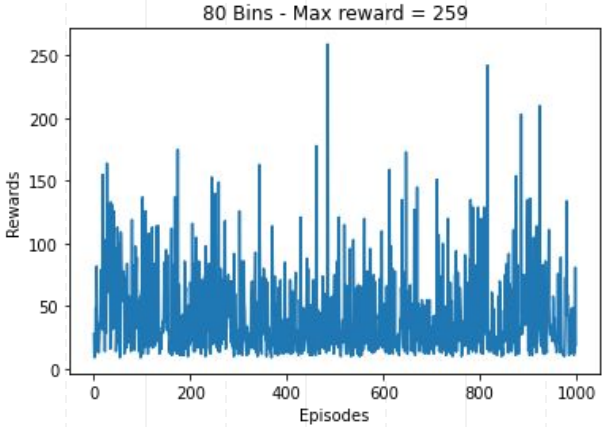
Some potential explanations for this:

- There truly is no correlation.
- There is only a correlation for environments with more randomness like the Atari environments.

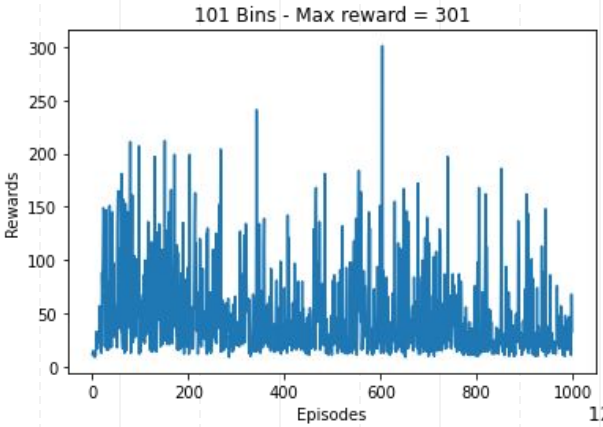
Results – C51

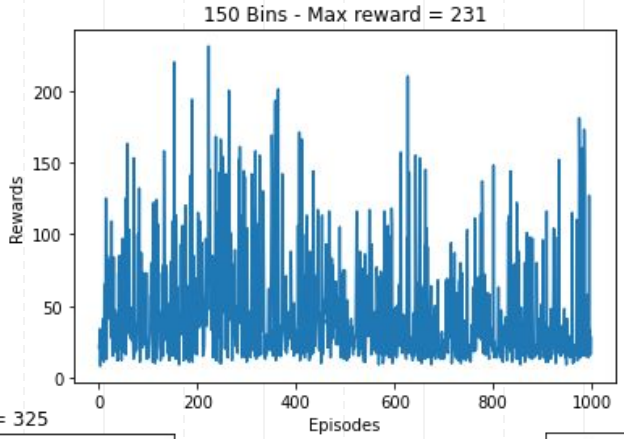

21 Bins - Max reward = 219
Rewards
Episodes
0
50
100
150
200
0
200
400
600
800
1000

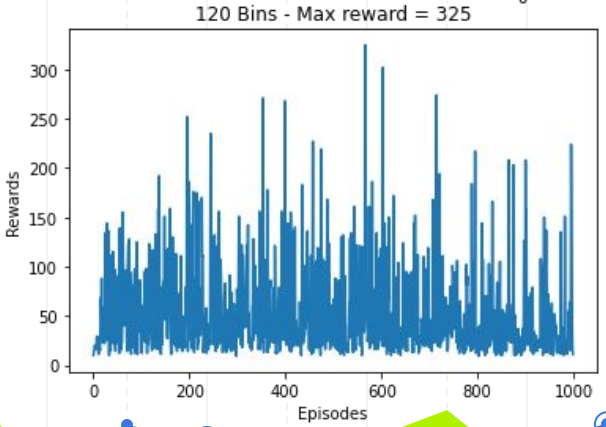

51 Bins - Max reward = 275
Rewards
Episodes
0
50
100
150
200
250
0
200
400
600
800
1000

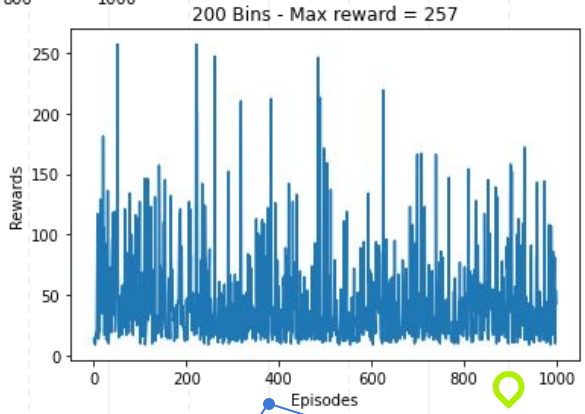

80 Bins - Max reward = 259
Rewards
Episodes
0
50
100
150
200
250
0
200
400
600
800
1000

Results – C51


101 Bins - Max reward = 301
Rewards
Episodes
0
50
100
150
200
250
300
0
200
400
600
800
1000

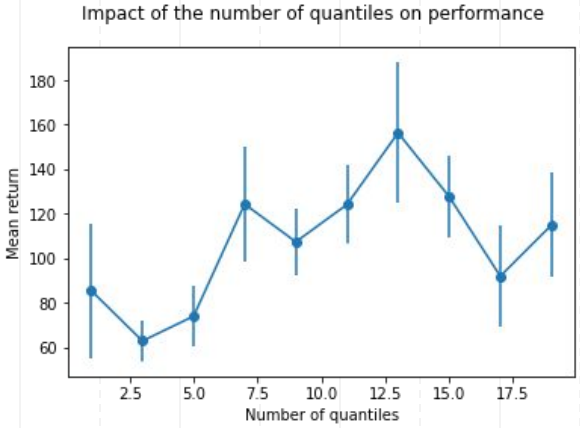

150 Bins - Max reward = 231
Rewards
Episodes
0
50
100
150
200
0
200
400
600
800
1000


120 Bins - Max reward = 325
Rewards
Episodes
0
50
100
150
200
250
300
0
200
400
600
800
1000


200 Bins - Max reward = 257
Rewards
Episodes
0
50
100
150
200
250
0
200
400
600
800
1000

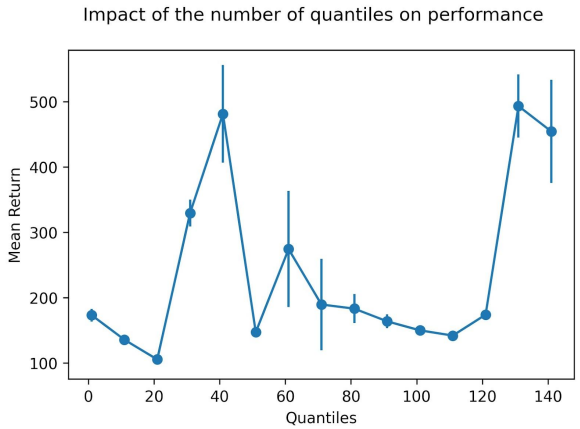
Results – QR-DQN

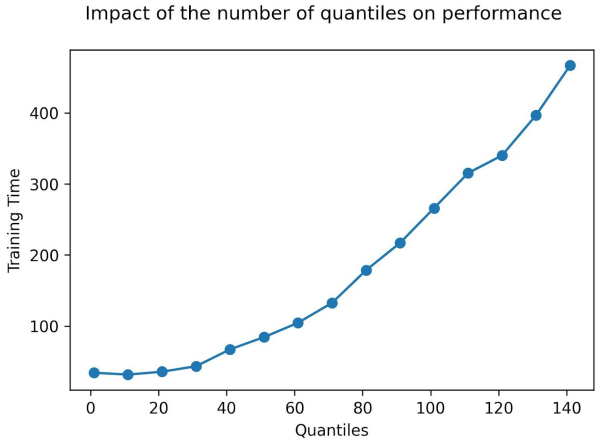
- Results averaged over 10 iterations for each number of quantiles
- Perhaps a slight upward trend but nothing significant



Results – QR-DQN

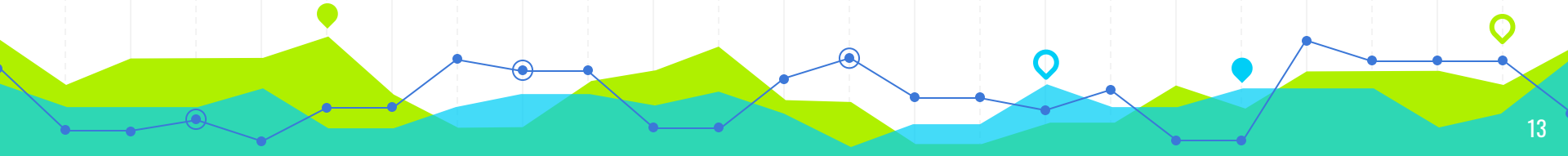
- No averaging, but much finer partitioning of quantiles
- Perhaps a slight upward trend but nothing significant
- Computation time goes up (as expected)





Future Work

- Conducting the same experiments in Atari environments
- Comparison of IQN and QR-DQN with high number of quantiles
- More ways to do distributional reinforcement learning?
  - Partition based on deviation from mean?



THANK YOU!

Any questions?