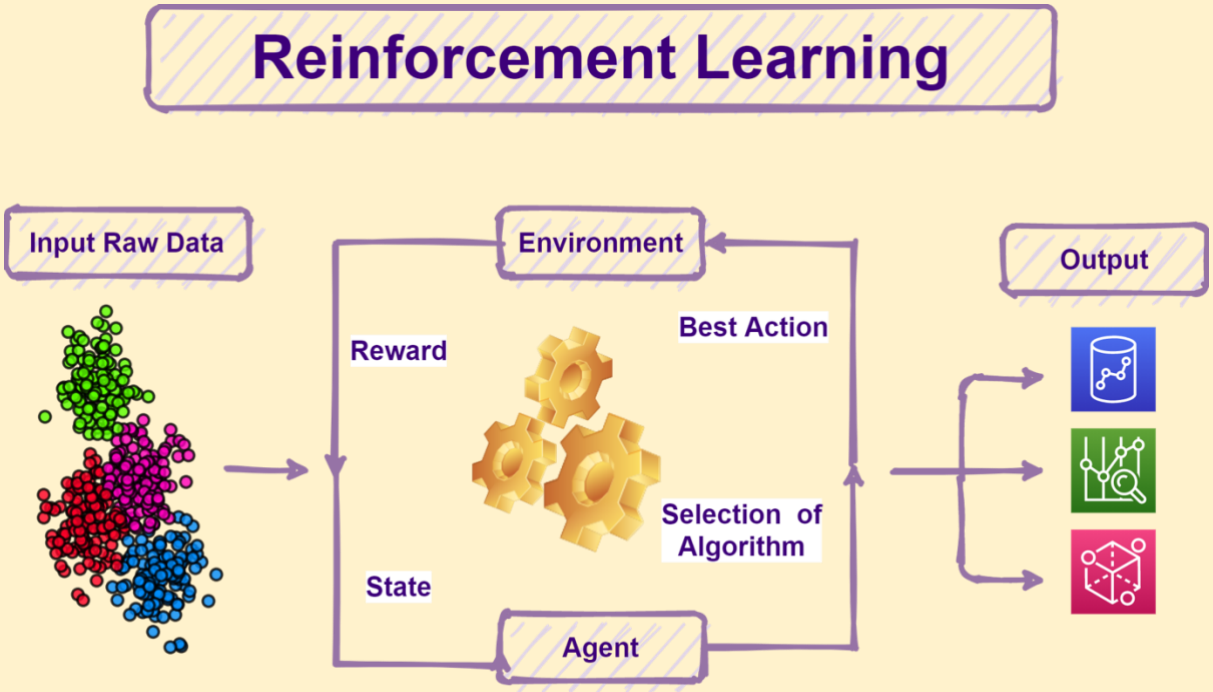
Reinforcement Learning for the CartPole Control Problem with a Simple Feed Forward Neural Network and Annealed Epsilon-Greedy Policy

A Critical Review



Introduction

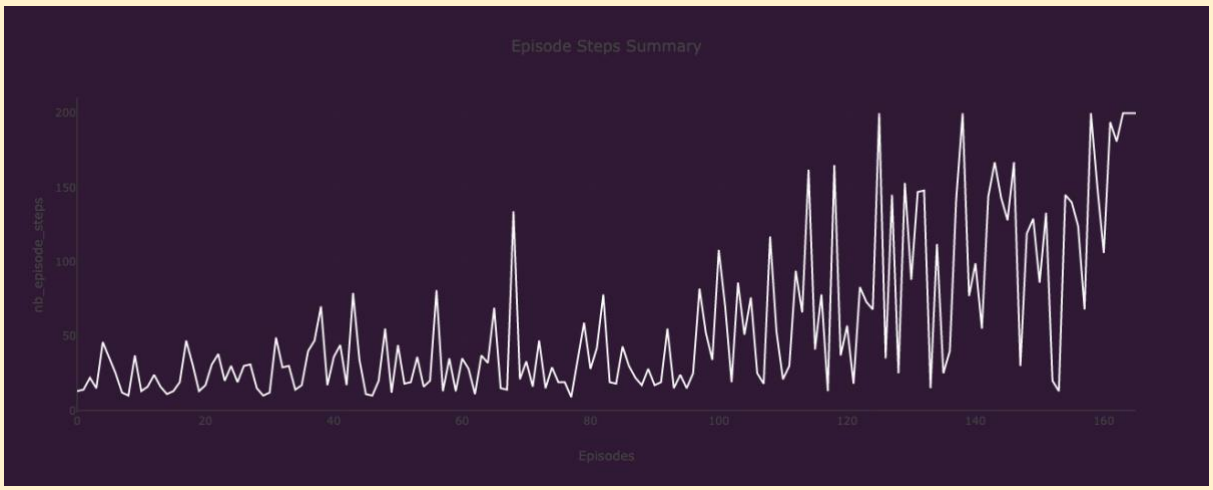
We use DQN in deep reinforcement learning to solve the CartPole-v0 challenge, a benchmark problem that involves balancing a pole on a moving cart with two control inputs.

Model Used

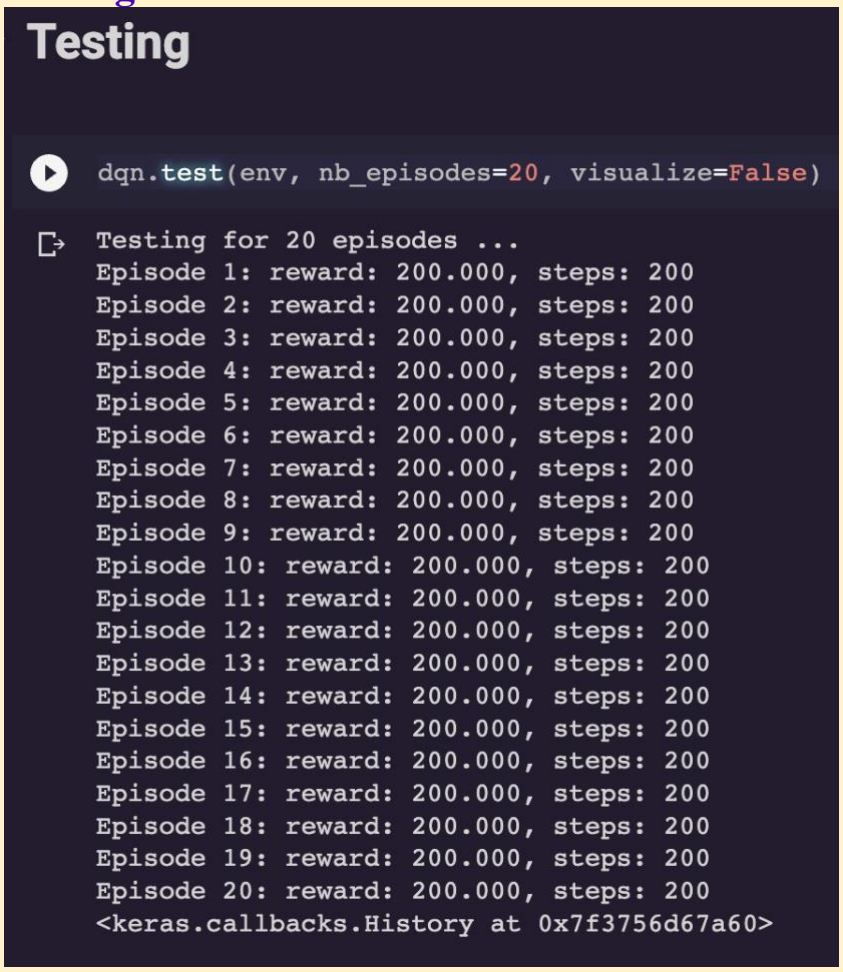
```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_4 (Flatten)         (None, 4)                 0

 dense_12 (Dense)            (None, 64)                320

 dense_13 (Dense)            (None, 16)                1040

 dense_14 (Dense)            (None, 2)                 34

=================================================================
Total params: 1,394
Trainable params: 1,394
Non-trainable params: 0
_________________________________________________________________
```

Number of Steps

```
 7710/9500: episode: 155, duration: 1.056s, episode steps: 145, steps per second: 137, episode reward: 145.000,
 7850/9500: episode: 156, duration: 1.011s, episode steps: 140, steps per second: 138, episode reward: 140.000,
 7974/9500: episode: 157, duration: 0.951s, episode steps: 124, steps per second: 130, episode reward: 124.000,
 8042/9500: episode: 158, duration: 0.529s, episode steps:  68, steps per second: 128, episode reward: 68.000,
 8242/9500: episode: 159, duration: 1.488s, episode steps: 200, steps per second: 134, episode reward: 200.000,
 8392/9500: episode: 160, duration: 1.272s, episode steps: 150, steps per second: 118, episode reward: 150.000,
 8498/9500: episode: 161, duration: 0.779s, episode steps: 106, steps per second: 136, episode reward: 106.000,
 8692/9500: episode: 162, duration: 1.435s, episode steps: 194, steps per second: 135, episode reward: 194.000,
 8873/9500: episode: 163, duration: 1.726s, episode steps: 181, steps per second: 105, episode reward: 181.000,
 9073/9500: episode: 164, duration: 1.504s, episode steps: 200, steps per second: 133, episode reward: 200.000,
 9273/9500: episode: 165, duration: 1.439s, episode steps: 200, steps per second: 139, episode reward: 200.000,
 9473/9500: episode: 166, duration: 1.525s, episode steps: 200, steps per second: 131, episode reward: 200.000,
done, took 77.093 seconds
```



Testing



Methodology

The model architecture consists of a Flatten layer, two Dense layers (with 64 and 16 units), and a Rectified Linear Unit activation function. The final Dense layer also utilizes ReLU activation to generate Q-value estimates for each action. The agent's exploration is enabled by an EpsGreedyQPolicy object, while a LinearAnnealedPolicy object facilitates the decay schedule for epsilon-greedy. The SequentialMemory module is used to store the agent's experience. The agent is trained for 9500 time steps, and its performance is evaluated based on 20 episodes.

Critique

To improve the agent's long-term learning capacity while minimizing the number of training steps, adjustments were made to several parameters. Specifically, the memory limit was increased to 1000, the warmup parameter was raised to 45, and the number of model neurons was minimized to optimize performance. The agent's optimal learning rate was determined to be 0.01, while the Adam optimizer's default value of 0.001 was found to be suitable. Reducing the number of training steps below 9500 resulted in a decrease in the reward from 200. Further research may explore alternative policies or neural network architectures that could potentially reduce the number of required training steps for optimal performance.

Conclusion

In conclusion, the agent's long-term learning capacity was enhanced while minimizing the number of training steps by adjusting several parameters. The memory limit was increased, the warmup parameter was raised, and the number of model neurons was minimized. Future research could investigate alternative policies or neural network architectures that could potentially reduce the number of required training steps for optimal performance.