Reinforcement Learning Coursework 2

Name: Kangle Yuan

CID: 02472956

Department: Computing

Course: MSc Computing (Visual Computing and Robotics)

Question 1:

1.1 Hyperparameter Table

Hyperparameter	Value	Justification
Learning Rate for Optimizer	0.001	0.001 is a common-used learning rate, generally small enough to ensure a gradual and stable learning.
REPLAY_BUFFER_SIZE	10000	10000 is a large buffer size to store enough experiences for agent to learn. This helps with learning stability and optimization. The usage of memory is still affordable.
BATCH_SIZE	32	Using batch sampling can increase the learning efficiency, and 32 equips with a good balance among learning speed, sampling randomness, and computational efficiency.
EPSILON_START, EPSILON_END, EPSILON_DECAY (ε-greedy policy)	Start at 1.0, decay to 0.01 over 1000 episodes	A high epsilon value (1.0) encourages exploration initially, because the agent has no knowledge of the environment. Gradually decaying epsilon to 0.01 over 1000 episodes (in total 3000 episodes) ensures a gradual shift from exploration to exploitation.
TARGET_UPDATE_FREQUENCY	Update every 10 seconds	Updating the target network every 10 seconds balance the stabiiity and efficiency. Updating too frequently (eg. 1 second) can lead to instability, while infrequent updates (eg. 300 seconds) can slow down learning.
DQN([4, 64, 64, 2])	Two hidden layers, each with 64 neurons	2 layers with 64 neurons per layer ensures decent complexity, avoiding overfitting for the CartPole problem. This architecture is enough for learning the basic patterns of dynamics.
Optimizer	Adam	The Adam optimizer outperforms SGD because of the attribute of dynamic learning rate, which also faciliates quicker convergence.
NUM_RUNS = 10 EPISODE_PER_RUN = 300	10 runs with 300 episode per run	This architecture is computationally enough to produce a satisfied result.

Table 1 Hyperparameter Table

1.2 Learning Curve

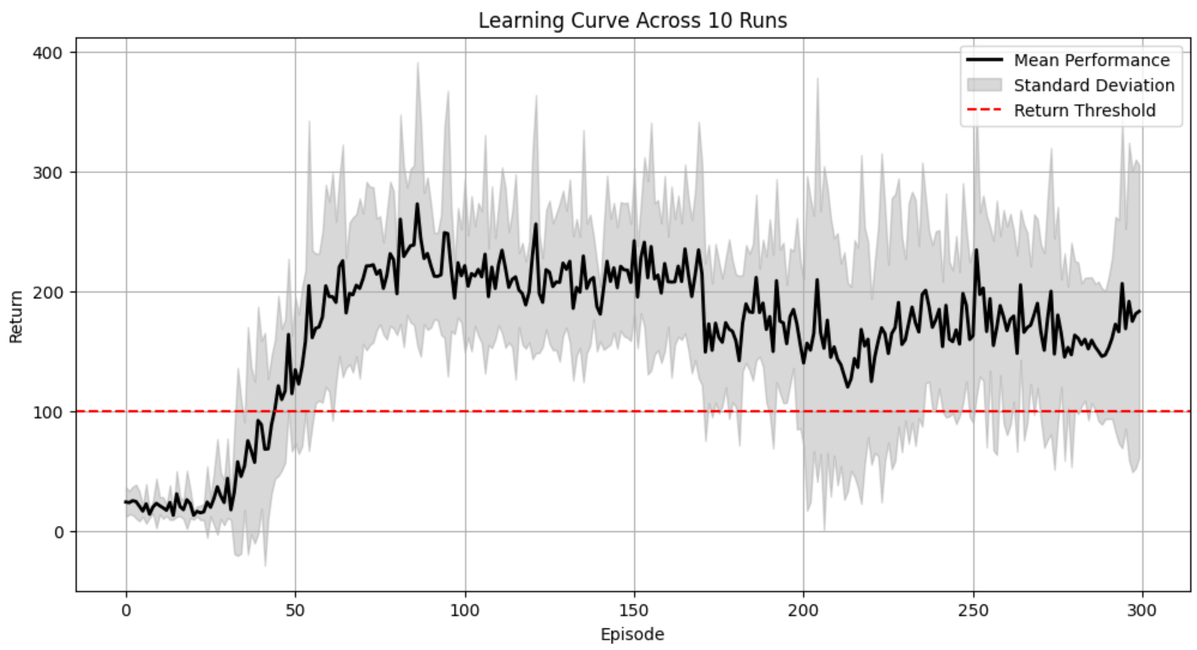

Figure 1 Learning Curve Across of DNQ Agent Across 10 Runs Plotted as Mean Return per Episode

Figure 1 shows that the mean return increases rapidly around 50 episodes, because the agent starts without knowledge, and the greedy policy gives the high exploration rate. The DQN agent initially learns and improves its policy. The fluctuation of the mean return is less significant beyond 50 episodes. However, even though this DQN model qualifies as a “successful” model after 50 episodes, there is a wide standard deviation across 10 runs, indicating that the current policy has not yet converged to stability. The randomness introduced by epsilon and sampling could be the reason for this. In conclusion, there is definitely room for improvement that could be executed here to further optimize the model, aiming for better performance.

Question 2:

2.1 Greedy Policy Action

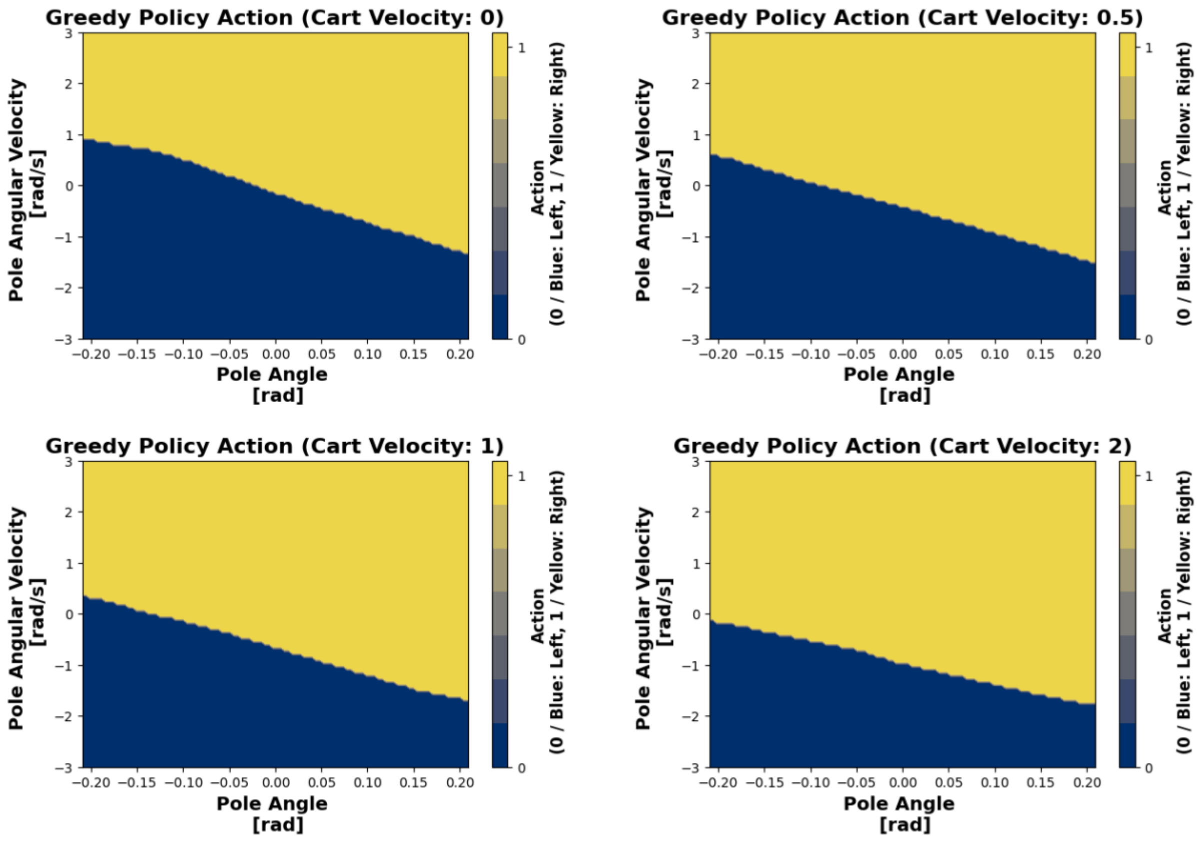

Figure 2 Greedy Policy Actions of a Trained DQN Agent at Velocities 0, 0.5, 1 and 2

Overall, Figure 2 indicates that the trained DQN agent has a clear and consistent decision boundary along the x-axis between yellow (right) and blue (left), across these four velocities. This suggests that the agent has learned a distinct strategy for balancing the pole. When the pole angle is negative, the agent tends to push left (blue), and when positive, to push right (yellow). This aligns with the strategy of pushing the cart in the direction the pole is falling.

Moreover, the decision boundary across the four plots has a similar diagonal orientation. When the angular velocity is zero, the agent is more sensitive to any deviations, adjusting the position to keep the pole vertical. The diagonal trend also shows that the agent tends to support one consistent action, primarily pushing in a direction to save the pole's fall.

The graphs clearly demonstrate that the decisions made by the agent are barely influenced by the cart velocity. This suggests that the policy has the potential to generalize well to different cart velocities. However, there may be an opportunity for fine-tuning to figure out the correlation furthermore.

2.2 Q-Value

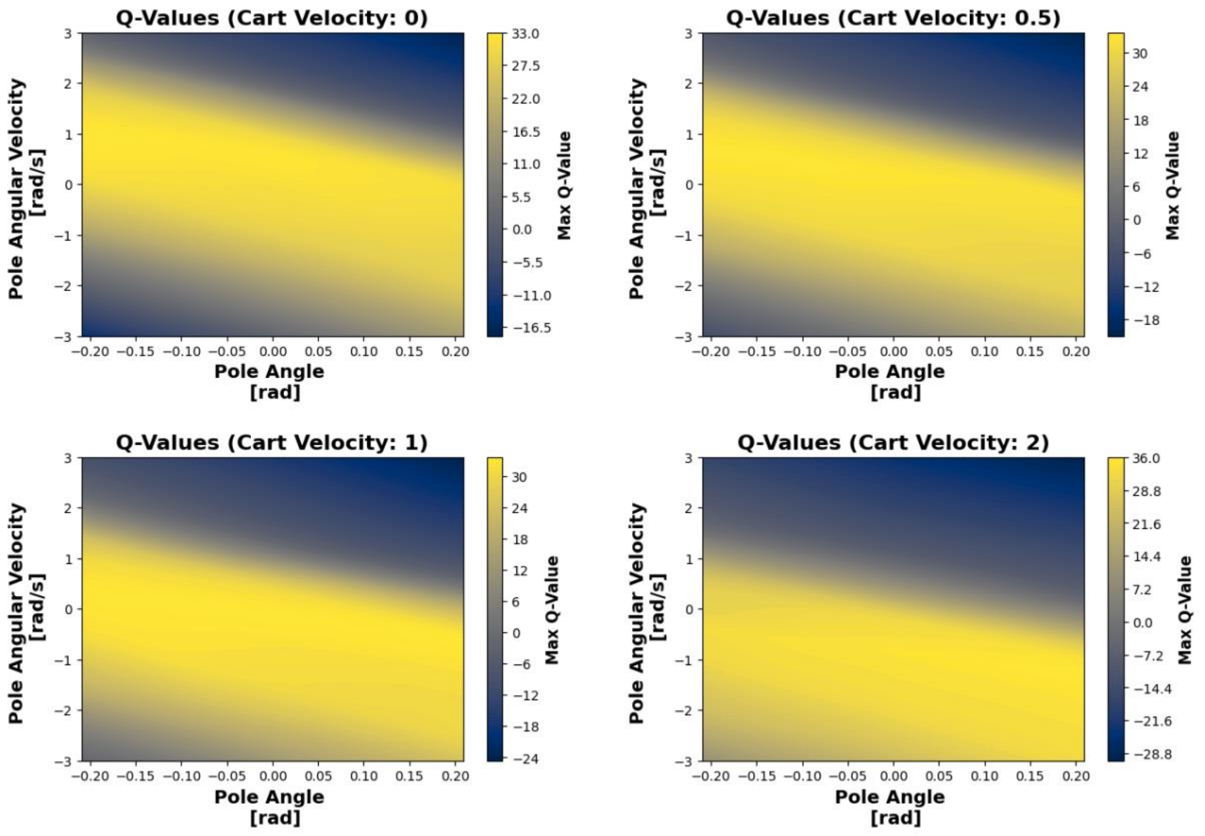

Figure 3 Maximum Q-values Learned by a DQN Agent for Different Cart Velocities at 0, 0.5, 1, and 2

Figure 3 displays a colour gradient transitioning from blue to yellow, representing varying expected return across different states. A consistent pattern is observed here, where higher values are assigned to states, where the pole angle is near zero and angular velocity is minimal. This aligns with the game's objective of keeping the pole upright.

However, despite the similarity in the pattern, the intensity of these Q-values varies. For instance, when the cart velocity reaches 2, the yellow region (higher Q-values) expands and become more obvious than when the cart is stationary or moving slowly. This suggests that higher velocities might associated with higher maximum Q-values. These differences indicate a correlation between the state values and the cart's current velocity.

Moreover, the lower Q-values present at the extremes of the pole angle and angular velocity. This is the signal of increased difficulty of achieving balanced position from these states, as they are near the point where the pole is likely to fall over and then terminate the game.