Assignment 5

Title: Deep Q-Network (DQN) Hyperparameters Analysis

Name: Brennen Cramp

Date: 12/2/2025

Introduction

Deep Q-Network (DQN) learning is a reinforcement learning technique that uses deep neural-networks to calculate the optimal action-value function in complex environments which will allow agents to learn directly from inputs such as hyperparameters. For the experimentation of hyperparameters in a DQN, the car-in-a-grid code (car_dqn_visualizer.py) will be leveraged to examine the effects of changing the values for the hyperparameters. The key hyperparameters will be adjusted to observe their impact on the learning process and final policy. The goal is to develop an understanding of the critical aspects of DQN such as the exploration-exploitation trade-off (epsilon; ε) and the planning horizon (gamma; γ).

Epsilon Decay Analysis

The epsilon decay value controls how quickly the agent switches from choosing actions randomly in a more explorative manor versus choosing actions based on the maximum Q-value calculated, in a more exploitative manor. The rate should be experimented at which the agent shifts from random exploration to exploiting using its learned knowledge and how that affects the optimum path found. For this section of the experiment, other hyperparameters will be the control variables and will not change from their default values (seen in *Table 1*) while the epsilon decay, which can be seen in *Table 2*, will be adjusted from high (5000) to medium (2500) to low (1000).

Parameter	Value	Note
Discount Factor (γ)	0.9 (Lock)	Must be constant for all Part 1 runs.
Learning Rate (α)	0.0005 (Baseline Value)	Must be constant for all Part 1 runs.
Epsilon Start (ε-start)	1	Starts fully exploring.
Epsilon End (ε-end)	0.05	Ends with 5% chance of random action.

Table 1: Control Hyperparameter Values for Epsilon Decay

Case	Epsilon Decay Rate	Strategy	Expected Outcome
1A	High (5000)	Rapidly favors exploitation.	Settles quickly, but potentially sub-optimally.
1B	Medium (10000)	Balanced.	Finds a solid path relatively efficiently.
1C	Low (15000)	Slowly favors exploitation.	Takes the longest to stabilize, but finds the optimal path.

Table 2: Epsilon Decay Experimental Values

After running the simulations, the high epsilon decay rate of 5000 (seen in *Figure 1*) illustrates the expected outcome that the decay plateaus relatively quickly and maintains that average reward with small fluctuations. The general curve for the medium epsilon decay rate of 10000 (seen in *Figure 2*) shows relatively typical and consistent fluctuation of the average reward time per episode throughout the entire run; also finishing in fewer episodes than the high epsilon decay rate where the medium decay rate finished in 574 episodes versus the high decay rate of 600 episodes. The low epsilon decay rate of 15000 (seen in *Figure 3*) also illustrates the expected outcome since it is supposed to take the longest to stabilize but finds the optimal path. The low epsilon decay rate also took the longest to complete with 1523 episodes as seen in *Figure 4* which contains all the decay rates for comparison, illustrating just how long it took to train itself

to find the most optimal path. Due to this, the medium decay value seems to be the most optimal by balancing exploration and exploitation instead of fully leaning into one or the other like the high and low epsilon decay values displayed, along with the quickest time it took to find the goal 10 times.

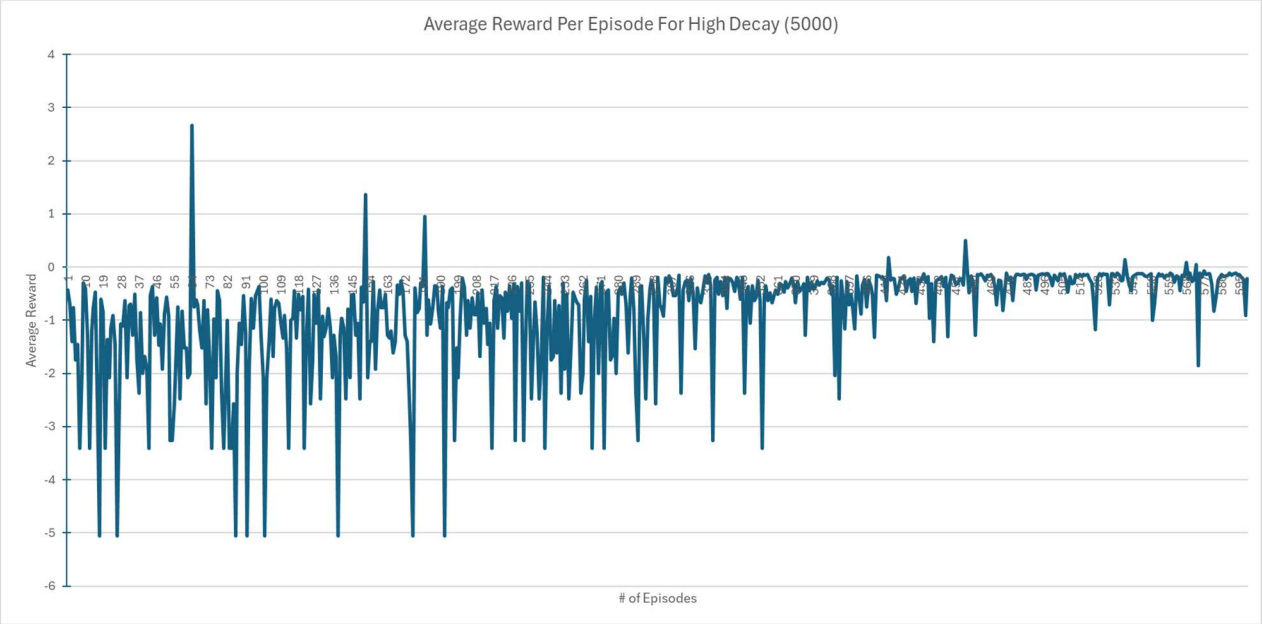

Figure 1: Average Reward Per Episode for High Decay (5000)

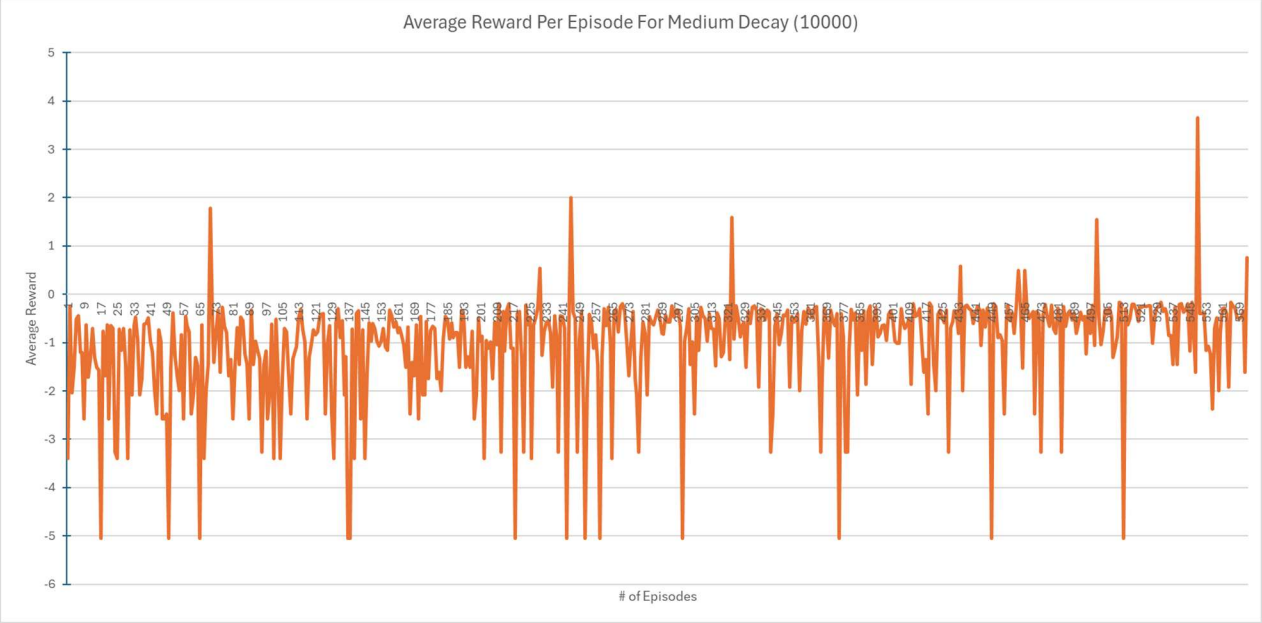

Figure 2: Average Reward Per Episode for Medium Decay (10000)

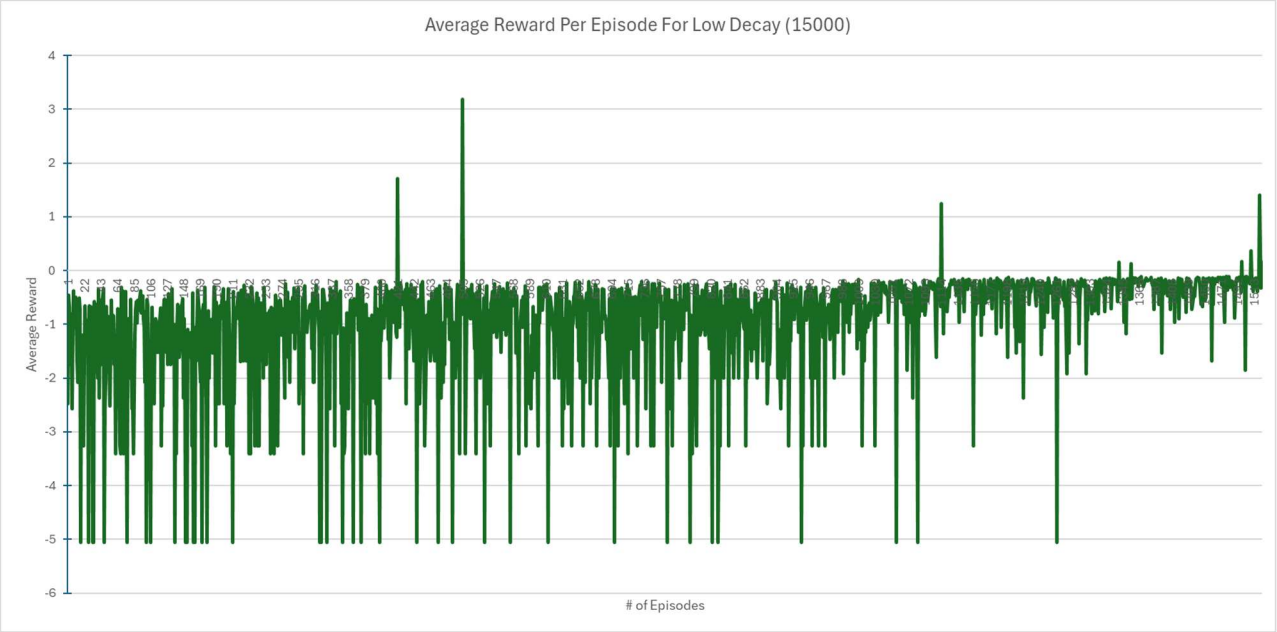

Figure 3: Average Reward Per Episode for Low Decay (15000)

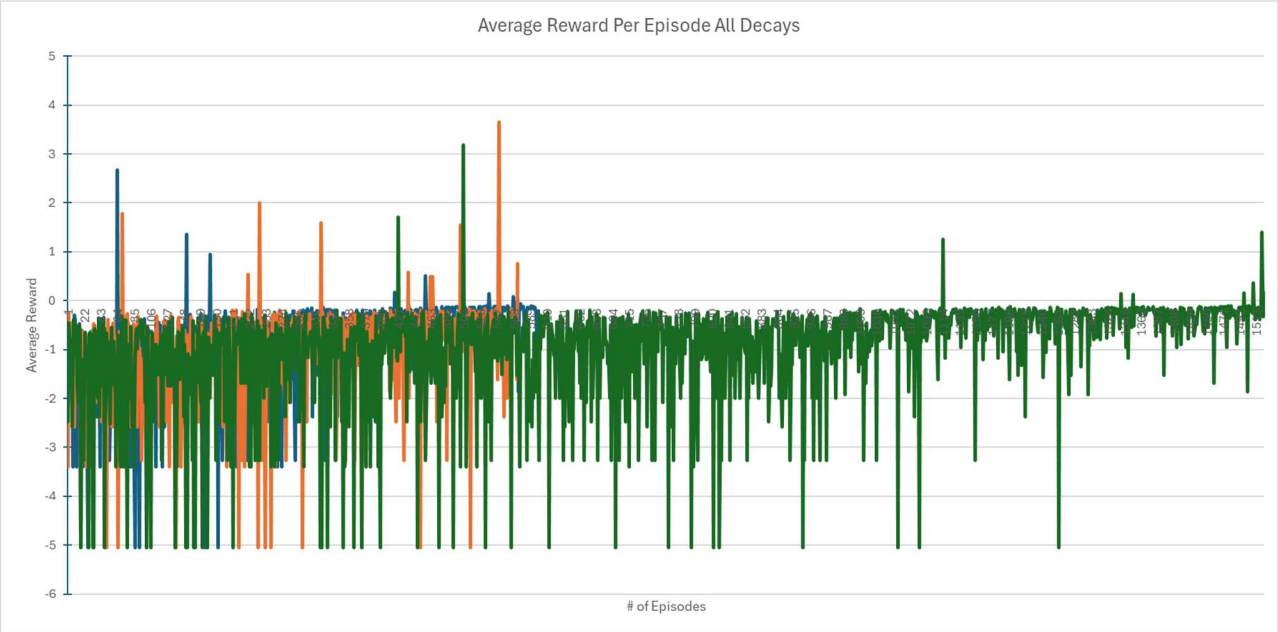

Figure 4: Average Reward Per Episode for All Decays

The Planning Horizon

The discount factor (gamma) determines the importance of future rewards versus the immediate rewards where a value closer to 1 makes future rewards (far-sighted) more important and a value closer to 0 makes immediate rewards (short-sighted) more important, making gamma the agent's

planning horizon. For this experiment, gamma is sought as the next hyperparameter to adjust. Like part 1 of the experiment, other hyperparameters will be the control variables and will not change from their default values (seen in *Table 3*) while the gamma, which can be seen in *Table 4*, will be adjusted from low (0.1) to medium (0.5) to low (0.99). By investigating this hyperparameter's range of values, it can help explore how the agent weighs immediate costs of R = -1 against the distant goal reward of R = +10.

Parameter	**Value**	**Note**
Epsilon Decay Rate (ε)	10000 (Lock)	Must be constant for all Part 2 runs. The most successful decay from Part 1.
Learning Rate (α)	0.0005 (Baseline Value)	Must be constant for all Part 2 runs.
Epsilon Start (ε-start)	1	Starts fully exploring.
Epsilon End (ε-end)	0.05	Ends with 5% chance of random action.

Table 3: Control Hyperparameters for Gamma

Case	**Gamma**	**Interpretation of Gama**	**Expected Outcome**
2A	Low (0.1)	Short-sighted.	Finds a longer than optimal path.
2B	Medium (0.5)	Balanced.	Good performance.
2C	High (0.99)	Far-sighted.	Finds the shortest path.

Table 4: Gamma Discount Factor Experimental Values

After running the simulations, the low gamma value of 0.1 (seen in *Figure 5*) illustrates the expected outcome that the discount factor being low means the agent is very short-sighted, meaning the goal reward is discounted with each step. The longest path took 678 steps and the shortest path took 15 steps with 1556 episodes where the car appeared to be turning in circles at points. This means the agent is not really motivated to quickly move toward the goal, leading the agent to find longer than optimal paths or even appear stagnant if a local state offers less of a

penalty than as a necessary step to find a better path. For the medium gamma value of 0.5 (seen in *Figure 6*), it shows a more efficient search where it leverages goal and immediate rewards and a reduced episodic loop with 1019 episodes with the longest path being 366 steps and the shortest path being 17 steps. Noticeably, the longest path has been reduced by more than 300 steps from the low gamma episodic loop but the shortest path has increased by 2 steps. This could mean the short-sighted agent got lucky and stumbled into the goal state while the medium discount factor took both goal and immediate rewards into consideration to find the goal. Finally, the far-sighted high gamma value of 0.99 (seen in *Figure 7*), where it is focused heavily on the goal reward, also met its expected outcome by finding the shortest possible path of all the discount factors with 14 steps, the smallest longest possible path with 146 steps, and the shortest episodic loop of 980 episodes. This shows that when an agent is extremely goal-oriented, it will have an average of the shortest path traversal and time it takes to reach the goal 10 times.

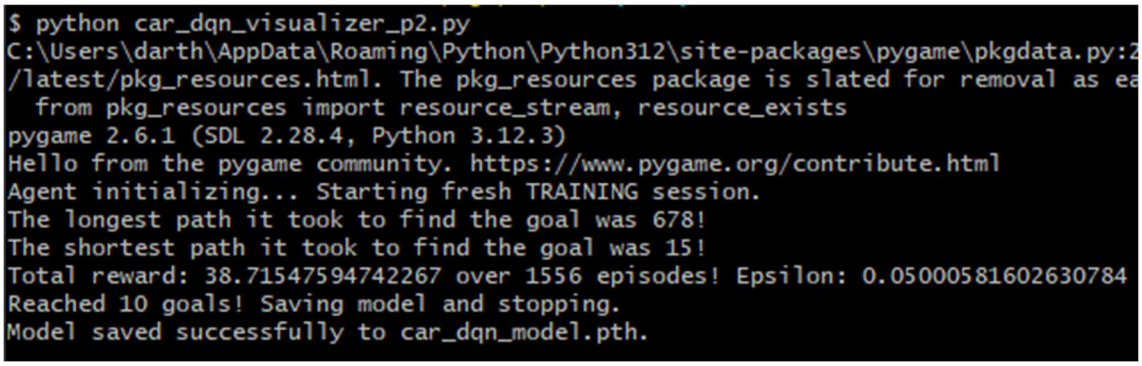

Figure 5: Low Gamma Discount Factor Min/Max Paths to Goal Terminal Output

```
$ python car_dqn_visualizer_p2.py
C:\Users\darth\AppData\Roaming\Python\Python312\site-packages\pygame\pkgdata.py:2
/latest/pkg_resources.html. The pkg_resources package is slated for removal as ea
  from pkg_resources import resource_stream, resource_exists
pygame 2.6.1 (SDL 2.28.4, Python 3.12.3)
Hello from the pygame community. https://www.pygame.org/contribute.html
Agent initializing... Starting fresh TRAINING session.
length of array: 10
The longest path it took to find the goal was 366!
The shortest path it took to find the goal was 17!
Total reward: 50.81547594742265 over 1019 episodes! Epsilon: 0.05859913816266889
Reached 10 goals! Saving model and stopping.
Model saved successfully to car_dqn_model.pth.
```

Figure 6: Medium Gamma Discount Factor Min/Max Paths to Goal Terminal Output

```
$ python car_dqn_visualizer_p2.py
C:\Users\darth\AppData\Roaming\Python\Python312\site-packages\pygame\pkgdata.py:
/latest/pkg_resources.html. The pkg_resources package is slated for removal as e
  from pkg_resources import resource_stream, resource_exists
pygame 2.6.1 (SDL 2.28.4, Python 3.12.3)
Hello from the pygame community. https://www.pygame.org/contribute.html
Agent initializing... Starting fresh TRAINING session.
The longest path it took to find the goal was 146!
The shortest path it took to find the goal was 14!
Total reward: 37.91547594742268 over 980 episodes! Epsilon: 0.09707593727499883
Reached 10 goals! Saving model and stopping.
Model saved successfully to car_dqn_model.pth.
```

Figure 7: High Gamma Discount Factor Min/Max Paths to Goal Terminal Output

Conclusion

In essence, the extremes of both the epsilon decay rate and the gamma discount factor show how the fluctuation of those hyperparameters affects the delicate nature of the DQN. If the epsilon decay rate is changed heavily to skew towards either end of the range of values, it can cause the DQN to settle on a sub-optimal path due to its heavy emphasis on exploitation by plateauing on where it believes the best path is as opposed to exploration by having long episodic loops with it can offer the best change of finding the optimal path but is very time consuming. If the gamma rate is changed heavily to skew towards either end of the range of values, it can cause the DQN to be “lost” by not being goal-oriented and just cares about what its next step is to being very goal-oriented, which as a positive, can lead to the shortest, most optimal path and in a short amount of time. Fully committing to either end of the spectrum should be utilized based on the

system. For epsilon decay rate, varying in the medium values appears to be the most optimal solution and for gamma discount function, leaning more towards goal-oriented solutions appeared to be optimal for this exercise but these values need to be carefully adjusted based on the requirements of the system where the DQN will be utilized.