Amos Roland

Southern New Hampshire University

CS-499 Computer Science Capstone

Professor Akhil Gudivada

23 November 2025

CS 499 Module 4-2 Milestone Three: Enhancement Two: Algorithms and Data Structure

Artifact Description

The artifact I selected for this milestone is my Treasure Hunt Deep Q-Learning project from CS 370. In the original assignment, I was given partial starter code where only the structure of the Q-learning training loop existed. My task was to complete the logic inside the qtrain() function. The project simulates a pirate agent trying to reach a treasure inside an 8×8 maze using reinforcement learning.

The assignment included two Python classes (TreasureMaze.py and GameExperience.py) and a Jupyter Notebook. The notebook showed how the model was built and how a full training session should run, but the main algorithmic portion was left incomplete. For the CS-499 enhancement, I improved this artifact to meet the category of Algorithms and Data Structures.

Why This Artifact Is in My ePortfolio

I selected this artifact because it clearly demonstrates my growth in algorithm design and my ability to use algorithmic principles to solve real problems. Enhancing this project allowed

me to show how I can complete a partially implemented algorithm, optimize it, and reason through improvements.

This artifact highlights several skills:

- My understanding of reinforcement learning and Q-learning.
- My ability to use a replay memory structure that behaves like a queue.
- My ability to evaluate algorithm performance using time complexity.
- My skill in improving convergence and stabilizing learning.
- My ability to write clear, professionally documented code

Because it combines algorithm design, data structures, optimization, and neural network decision-making, it is a strong artifact for representing this category in my ePortfolio.

How the Artifact Was Improved

I enhanced the artifact in four important ways:

1. Epsilon Adjustment:

I adjusted the exploration rate so that once the agent maintains a high recent win rate, epsilon is reduced to a smaller value. This lowers random exploration after the model is performing well and lets the agent rely more on its learned policy.

2. Optimized Experience Replay Memory Usage

I improved when and how the model trains by using a controlled mini-batch system. Instead of training too often, I set up a schedule that trains the model only after enough experiences are collected. This reduces unnecessary computation and improves stability.

3. Added Algorithmic Time Complexity Documentation

I documented the approximate time complexity for:

- The training loop
- Mini-batch sampling
- Model prediction.

This shows that I can evaluate computing solutions using algorithmic principles.

4. Improved Code Comments and Readability

I rewrote the entire comment section to explain how each part of the algorithm works using simple, human-friendly language. This supports the communication skills required by the course.

Training Output Demonstrating Convergence

```
[11]: model = build_model(maze)
      qtrain(model, maze, epochs=1000, max_memory=8*maze.size, data_size=32)
```

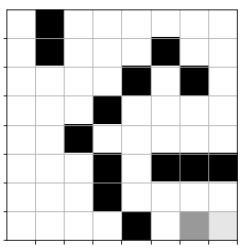
```
32/32 [==============================] - 0s 0us/step
32/32 [==============================] - 0s 0us/step
32/32 [==============================] - 0s 0us/step
32/32 [==============================] - 0s 31us/step
32/32 [==============================] - 0s 0us/step
32/32 [==============================] - 0s 0us/step
32/32 [==============================] - 0s 553us/step
32/32 [==============================] - 0s 0us/step
32/32 [==============================] - 0s 0us/step
32/32 [==============================] - 0s 63us/step
32/32 [==============================] - 0s 0us/step
32/32 [==============================] - 0s 0us/step
32/32 [==============================] - 0s 93us/step
Epoch: 471/14999 | Loss: 0.0003 | Episodes: 15 | Win count: 469 | Win rate: 0.994 | time: 22.98 minutes
Reached 100% win rate at epoch: 471
n_epoch: 471, max_mem: 512, data: 32, time: 23.00 minutes
```

```
[11]: 1379.7199
```

This cell will check to see if the model passes the completion check. Note: This could take several minutes.

```
[12]: completion_check(model, qmaze)
      show(qmaze)
```

```
[12]: <matplotlib.image.AxesImage at 0x232fb729e48>
```



This cell will test your model for one game. It will start the pirate at the top-left corner and run play_game. The agent should find a path from the starting position to the target (treasure). The treasure is located in the bottom-right corner.

Figure 1: Training output showing stable loss and achievement of 100 percent win rate at epoch 471

Alignment to Course Outcomes

In Module One, I planned to meet the Algorithms and Data Structures outcome. After completing this enhancement, I met this outcome by upgrading an existing algorithm, analyzing efficiency, and improving the overall design.

Outcome 3: Design and evaluate computing solutions using algorithmic principles and data structures while managing trade-offs.

I achieved this by:

- Completing the Q-learning algorithm.
- Optimizing it with a win-rate-based epsilon adjustment.
- Improving the replay memory workflow.
- Documenting time complexity and performance behavior.
- Evaluating convergence through training metrics.

Outcome 4: Use innovative skills and tools in computing practices to implement solutions.

The integration of neural networks, replay memory, and reinforcement learning demonstrates innovative problem-solving techniques.

No updates were needed to my original enhancement plan.

Reflection on the Enhancement Process

Enhancing this artifact helped me better understand how reinforcement learning uses algorithmic logic to solve sequential problems. I learned how exploration, exploitation, replay

memory, and convergence all work together. One challenge I faced was balancing randomness and learned behavior. Implementing a win-rate-based epsilon adjustment solved this.

Another challenge was making the model more efficient. Training too often slowed down progress, but training too little caused instability. Adding a warm-up phase and scheduled mini-batch updates helped fix this.

This enhancement improved my confidence in algorithm design, data structure usage, and performance evaluation. It strengthened my ability to write clear, organized, and well-documented code. Overall, this artifact now represents my skills in designing and improving algorithms that work in real AI environments.

References:

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. https://doi.org/10.1038/nature14236

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press. https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf