Project #3 - Reinforcement Learning

Hayden Moore (hmm5731@psu.edu)
Pattern Recognition and Machine Learning

1. **DQN vs. Q-Learning:**
   Compare results of traditional Q-Learning with DQN:

   First, lets get an idea of what each Algorithm is optimizing for.

   Q-Learning:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{1}$$

   Where:

   - $Q(s,a)$: Action-value for state s and action a
   - α: Learning rate $\in [0,1]$
   - r: Reward
   - γ: Discount factor $\in [0,1]$
   - s': Next state
   - $\max_{a'}(Q(s',a'))$: Maximum expected future reward

   DQN:

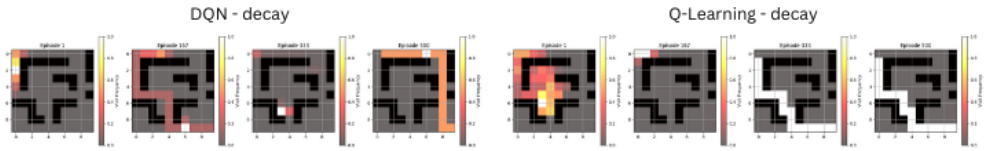
$$L(\theta) = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q_{\text{target}}(s',a') - Q_{\text{policy}}(s,a) \right)^2 \right] \tag{2}$$

   Where:

   - Q_{policy}: Active neural network
   - Q_{target}: Frozen target network
   - r: Reward
   - γ: Discount factor

   So for Q-Learning we use an epsilon-greedy approach where epsilon decays over time, reducing randomness as learning progresses. We then directly update a Q-table to learn the state space of the environment. This approach is typically more stable and structured in learning and finds a strong policy with steady updates. In our graph below for Q-Learning Decay, we see this initial randomness in the first heat map and then we see a strong policy that is learned in the last two heatmaps indicating that the agent has learned an effective path efficiently.

For DQN we use a neural network to approximate these Q-values but we still rely on this epsilon-greedy decay. DQN also needs experience replay and a target network to stabilize its learning, making it slightly more complex then Q-Learning but more scalable. In our graph below we see that DQN is able to find a policy to reach the end state but it is not as strong as the Q-Learning approach. This could be due to the complexity of the neural network incorrectly learning the simple state space of this 10x10 map.



2. **Comparing Exploration Strategies:**

Boltzmann: Here we assign probabilities to actions based on their Q-values using a softmax function, this can allow for more controlled and probabilistic exploration throughout the maze.

$$P(a_i) = \frac{\exp(Q(s, a_i)/\tau)}{\sum_j \exp(Q(s, a_j)/\tau)} \tag{3}$$

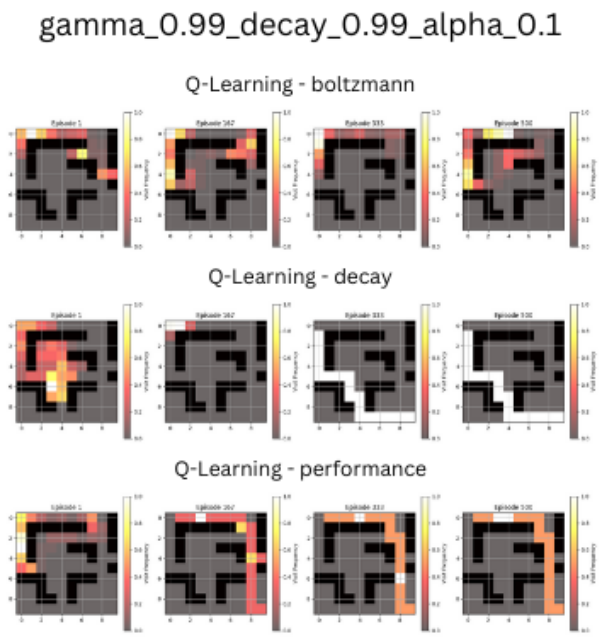
Decay: in this approach we gradually reduces the exploration rate over time using a decay factor, helping to ensure that the agent explores less as it learns more.

$$\epsilon_{t+1} = \max(\epsilon_{\min}, \epsilon_t \cdot \text{decay}) \tag{4}$$

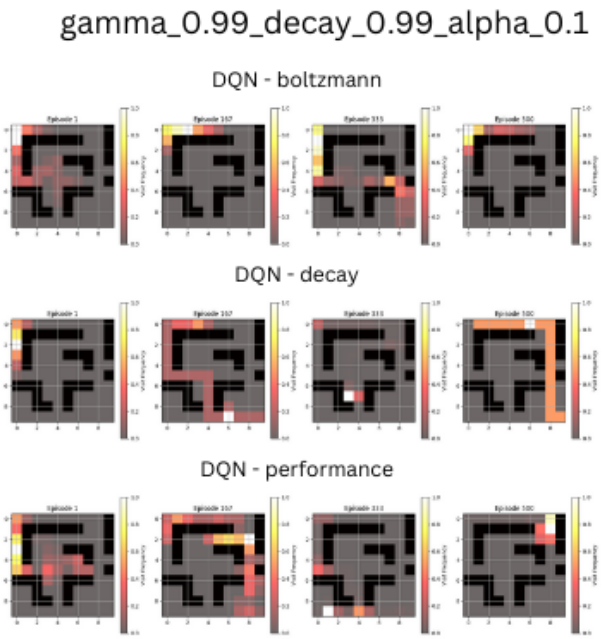
Performance Based: The agent selects a random action with some probability epsilon, and the best-known action with some probability 1 - epsilon. If the agent performs well, we can be reduce epsilon.

$$a_t = \begin{cases} \text{Random Action} & \text{with probability } \epsilon \\ \text{Best Known Action} & \text{with probability } 1 - \epsilon \end{cases} \tag{5}$$

Q-Learning 10x10 map: Boltzman, Decay, Performance Based *(alpha = 0.1, gamma = 0.99, decay_rate = 0.99)*:



DQN 10x10 map: Boltzman, Decay, Performance Based *(gamma = 0.99, decay_rate = 0.99)*:



In all scenarios from these few runs the Boltzman approach performed the worst. For both DQN and Q-Learning using the decay approach, the agent seemed to find a path to the goal, however Q-Learning seemed to learn a stronger path. Additionally, Q-Learning using the performance approach, the agent was able to find a decently strong path to the end state. It is also worth noting that different paths were found between

the DQN Decay vs Q-Learning Decay, indicating potential differences for how these approaches learn and showing that there are multiple solutions to these maze problems that can be learned from the agent.

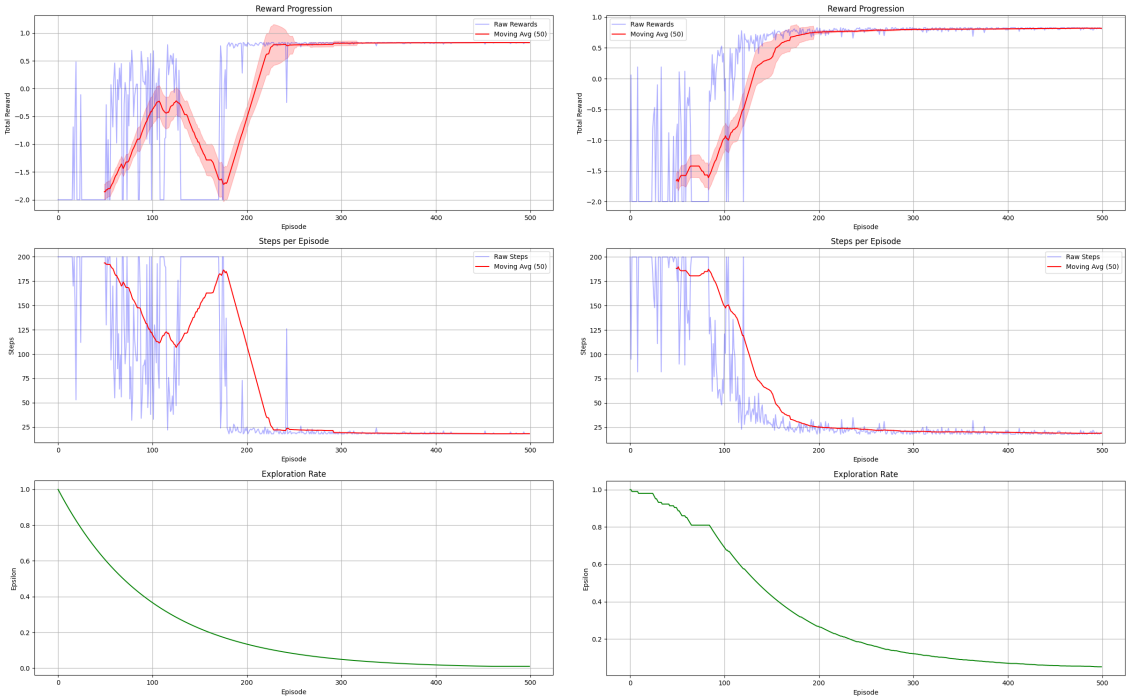
3. **Comparing results of different hyperparameters:**

Q-Learning 10x10 map:

Boltzman, Decay, Performance Based: *(alpha = 0.1, gamma = {0.5, 0.9}, decay_rate = {0.5, 0.9})*:

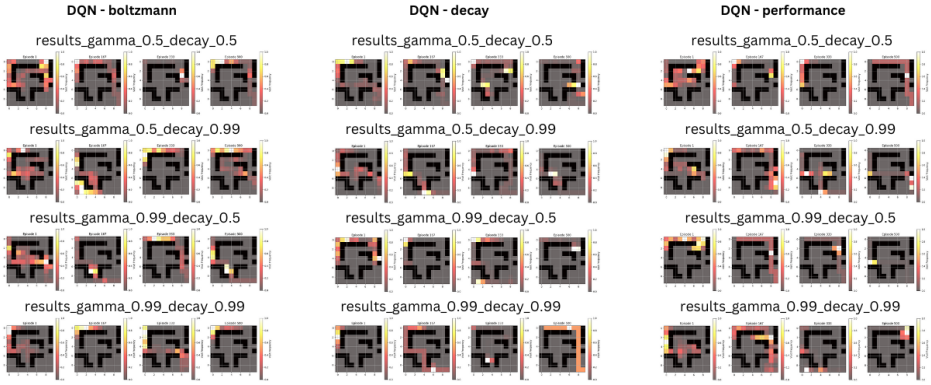
When looking into the metrics from these runs we can see a slight difference in some of the successfull approaches. When comparing Q-Learning Decay results alpha 0.1, gamma 0.99, decay 0.99 vs Q-Learning Performance results alpha 0.1, gamma 0.99, decay 0.99 we can see a slightly more stable Reward Progression for the Performance Based approach. Both approaches seem to have a pretty stable Exploration rate across the episodes with the Decay approach having slightly more stable performance.

Below is a comparison of Reward Progression & Exploration Rate for Q-Learning Decay results alpha 0.1, gamma 0.99, decay 0.99 vs Q-Learning Performance results alpha 0.1, gamma 0.99, decay 0.99.



DQN 10x10 map:

Boltzman, Decay, Performance Based: *(layer_depth = 2, gamma = {0.5, 0.9}, decay_rate = {0.5, 0.9})*:

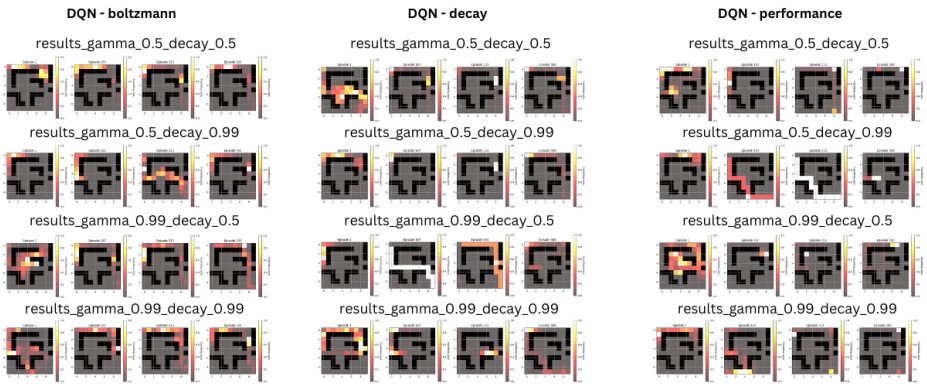


Boltzman, Decay, Performance Based: *(layer_depth = 4, gamma = {0.5, 0.9}, decay_rate = {0.5, 0.9})*:

By increasing the DQN Network layer depth from 2 to 4 we can improve the results for some of these runs. Specifically, if we look at the results from DQN decay results_gamma_0.99_decay_0.5 and DQN performance results_gamma_0.5_decay_0.99 we can see that these runs are able to solve the maze, where its 2 layer counterparts failed to find the end state.

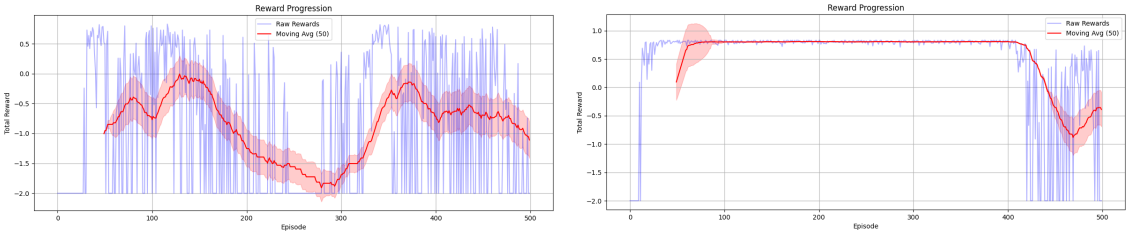
This could indicate that increasing the layer depth can help the network to capture better representations of the maze states. We are increasing the depth of the network

which also increases the total number of trainable parameters allowing for the network to potentially learn a better decision boundary and path to the end state.



It is also worth noting that the metrics clearly improve in some of these scenarios when we increase the network depth from 2 to 4 layers for DQN. By looking at the Reward Progression we can clearly see that the 2-layer approach is struggling to find consistent rewards hinting towards it being lost. Whereas, the consistent and high Reward Progression from the 4-layer approach more clearly indicates we have found a strong reward path to the goal.

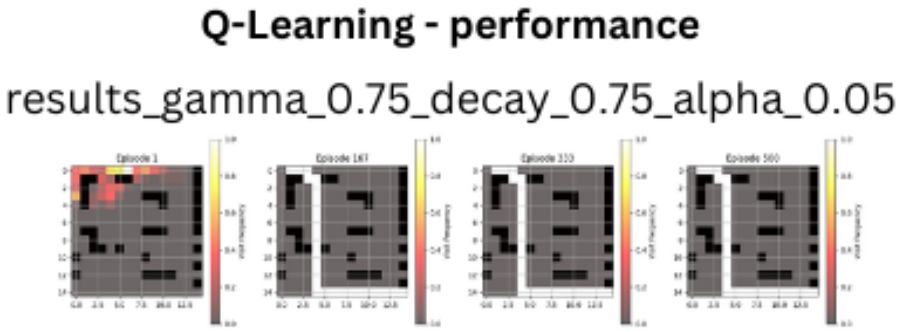
Below is a comparison of Reward Progression for DQN Decay results_gamma_0.99_decay_0.5 (2-layers) vs DQN Decay results_gamma_0.99_decay_0.5 (4-layers)



Increasing the complexity and size of the map, 10x10 map to 15x15 map:

Q-Learning 15x15: *(alpha = 0.05, gamma = 0.75, decay_rate = 0.75)*:

When I moved into the larger 15x15 sized map, most of our Q-Learning approaches failed to find the end goal. However, by using the performance based approach and balancing out the gamma to 0.75, decay_rate to 0.75, and alpha to 0.05 we were able to solve the 15x15 maze incredibly well. Seemingly, 0.75 was a good hyperparameter for the 15x15 map for the agent to consider future rewards and prevented excessive exploration. This is clearly shown in the second heatmap (Episode 180) we have already strongly found a path to the end goal. Also by dropping the learning rate from 0.1 to 0.05 which helped with steady updates to the Q-values, and prevented drastic shifts in learned values that could destabilize learning in the larger 15x15 state space.

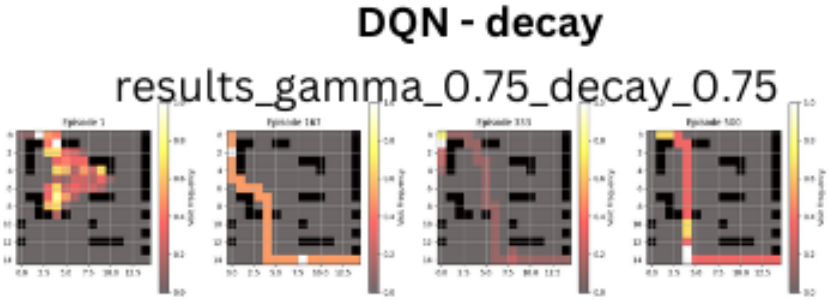


DQN 15x15: *(layer_depth = 4, gamma = 0.75, decay_rate = 0.75)*

When I moved into the larger 15x15 sized map, most of our Deep Q-Network (DQN) approaches struggled to find the end goal effectively. However, by using the decay approach and adjusting the gamma to 0.75 and the exploration decay rate to 0.75, I was able to improve performance. While these results were not as strong as with Q-Learning, they still demonstrated the importance of fine-tuning hyperparameters for larger state spaces.

Solving this for DQN required us to move from a 2-layer network to a four-layer network. This seemed to help the agent learn more complex representations of the environment. (Similar to our results with Q-Learning 15x15) The gamma value of 0.75 allowed the agent to consider future rewards while avoiding excessive exploration, helping it focus on meaningful long-term strategies. The decay rate of 0.75 ensured a gradual shift from exploration to exploitation, allowing the network to refine its policy effectively over time. A deeper architecture can provide better feature extraction capabilities but also increased training complexity, which could contribute to the slightly weaker performance compared to Q-Learning.

While DQN did not perform as strongly as Q-Learning in this scenario, the use of a four-layer architecture combined with careful hyperparameter tuning still significantly improved the agent's ability to solve the 15x15 maze.



Q-Learning 20x20:

I could not find any hyperparameters or approaches to solve a 20x20 maze with Q-Learning.

DQN 20x20:

I could not find any hyperparameters or approaches to solve a 20x20 maze with DQN. However, the results did slightly indicate that it was doing more exploration and recieving more rewards.

Overall I think that when we move into this larger 20x20 state space DQN is the appropriate approach to learn these complex features of the maze. However, this also may require delicate hyperparameter fine-tuning.

4. **Evaluating a Good Learning Algorithm:**
In the context of my experiments with Q-Learning and DQN, several key observations helped me to define what makes one algorithm superior to another for the different Maze scenarios.

Ability to Learn an Efficient Path:

A good algorithm for RL should be able to find the efficient path quickly without the need for extra overhead/compute. Depending on the environment and RL task some more complex solutions like Deep Layered DQN may solve the solution but much slower and more expensively than Q-Learning. However, in some scenarios we may need a Deeper DQN network to learn the complex state spaces of an environment.

Q-Learning showed stronger convergence in smaller mazes (10x10 and 15x15), particularly with the performance-based and decay exploration strategies. The Q-table representation allowed for direct updates, leading to a stable policy in these more simple state spaces.

DQN, however, struggled more with convergence, requiring an increase in the layer depth from 2 to 4 layers, which did improve its performance significantly, allowing it to learn better feature representations of the maze states. However, it is worth noting that the speed of convergence is much slower with this DQN approach compared to Q-Learning. It eventually learns the path but Q-Learning is able to learn a strong path much earlier in its training cycle.

Approaches that Balance their Exploration

A good exploration approach would likely try new actions to learn more about an environment but also choose the best-known action from previous experience. This is a very difficult thing to accomplish and all of the approaches we tested in this paper try to tackle this in different ways.

First, Boltzmann exploration performed the worst across both Q-Learning and DQN. The probabilistic action selection seemed to be too unstable, leading to excessive randomness and inefficient learning. I was unable to find a good balance of hyperparameters or map for this approach to excel.

Decay-based exploration performed well, especially in Q-Learning, by gradually reducing exploration and allowing the agent to refine its learned policy over time. This approach seemed to be the most flexible to differing hyperparameters compared to the other approaches.

Performance-based exploration yielded strong results in Q-Learning, allowing the agent to adaptively switch between exploration and exploitation based on its learning progress, specifically when challenged with the 15x15 map, this approach along with Q-Learning quickly learned a strong path to the end goal.

5. **Extra Credit:**
   N/A