Hyperparameter Impact on Learning Efficiency in Q-Learning and DQN using OpenAI Gymnasium Environments

***Abstract*:** This study compares the Q-learning and DQN methodologies within the CartPole-v1 environment. All methods were executed, trained, and evaluated according to test outcomes and training improvements. The research paper includes statistical summaries, and visualizations of performance and learning. This paper examines the impact of hyperparameters on the learning efficiency of Q-Learning and Deep Q-Network (DQN) in the CartPole-v1 environment of OpenAI Gymnasium. The results indicate that DQN substantially outperforms Q-Learning. DQN achieved a peak training reward of 500, whereas Q-Learning maintained an average test reward of 23. The hyperparameters that enhance DQN's neural network and performance include a learning rate of 0.001 and a batch size of 64. The efficacy of Q-Learning was hindered by inadequate hyperparameters and imprecise state discretization. The findings underscore the necessity of hyperparameter tuning to enhance the efficiency of reinforcement learning.

Keywords: *Reinforcement Learning, Q-Learning, Deep Q-Network (DQN), Hyperparameter Tuning, CartPole-v1*

I. INTRODUCTION

Reaching from robotics to game playing, reinforcement learning (RL) has evolved into a powerful paradigm for handling difficult sequential decision-making tasks. Under RL, an agent interacts with an environment under direction of rewards reflecting the desirability of its actions, so learning to make optimal decisions. For their capacity to address such tasks, two well-known RL algorithms Q-Learning and Deep Q-Network (DQN) have been extensively investigated. A model-free tabular method, Q-Learning depends on discretizing the state space to update a Q-table estimating the expected future rewards for state-action couples. DQN, on the other hand, uses deep neural networks to approximate Q-values, so allowing it to manage continuous state environments more successfully. But their hyperparameters learning rate, discount factor, exploration rate, and, in the case of DQN, neural network architecture and replay buffer size have a major impact on how these algorithms perform. Optimizing RL algorithms in practical uses depends on an awareness of how these hyperparameters affect learning efficiency.

Widely used toolkit for RL research, the OpenAI Gymnasium offers consistent settings such as CartPole-v1, a benchmark for assessing RL algorithms. CartPole-v1 requires the agent to learn a policy maximizing cumulative rewards by applying suitable forces as it involves balancing a pole on a moving cart. Based on past research as shown in the given results, DQN greatly beats Q-Learning in CartPole-v1, obtaining almost ideal test rewards of roughly 459 instead of Q-Learning's meager 23. These findings show how well DQN manages continuous state spaces using neural network approximations, while Q-Learning suffers with discretization restrictions. Nevertheless, the effectiveness of these methods evaluated in terms of convergence speed, stability, and final performance depends on well-tuned hyperparameters. Whereas the exploration rate (epsilon) balances exploration and exploitation, the learning rate (alpha in Q-Learning, optimizer rate in DQN) controls the step size of updates. Inappropriate hyperparameter values might cause slow convergence, instability, or inability to learn an efficient policy.

This research paper examines the influence of hyperparameters on the learning efficiency of DQN and Q-Learning in the CartPole-v1 environment. We seek to find configurations that maximize training and testing performance by methodically analyzing important parameters including alpha, gamma (discount factor), epsilon decay, and for DQN, batch size and target network update frequency. Building on earlier results including training and evaluation reward plots and summary statistics, the paper quantifies how hyperparameter changes affect reward trajectories and convergence rates. While DQN's training rewards can reach 500—a sign of an ideal policy—Q-Learning's rewards have plateaued at a low value, implying that the method's hyperparameter configuration or discretization technique has constraints. This work attempts to offer useful insights on hyperparameter tuning via controlled experiments, so improving the applicability of RL methods. Last but not least, the findings seek to help the RL community overall by clarifying how hyperparameter choices affect learning efficiency in benchmark settings, so producing more strong and scalable RL solutions.

II. Literature Review

A. Q-Learning and Its Hyperparameters

Q-Learning, introduced by Watkins and Dayan in 1989, is a model-free RL algorithm that learns a Q-table to estimate the expected future rewards for state-action pairs [14]. The algorithm updates the Q-values using the Bellman equation, with hyperparameters such as the learning rate (alpha), discount factor (gamma), and exploration rate (epsilon) playing critical roles. Shalika [8] provides a beginner's guide to Q-Learning, emphasizing that alpha controls the extent to which new information overrides old Q-values, while gamma determines the importance of future rewards. A high alpha may lead to unstable learning, whereas a low alpha can slow convergence. Similarly, gamma values close to 1 prioritize long-term rewards, potentially causing slower learning in environments with sparse rewards [13].

The exploration-exploitation trade-off, managed by epsilon in an epsilon-greedy policy, is another critical factor. Shalika [8] highlights that a high initial epsilon encourages exploration but must decay appropriately to ensure exploitation of learned policies. Improper epsilon decay rates can trap the agent in suboptimal policies or prevent sufficient exploration, as discussed by Rutherford et al [15] in the context of multi-agent RL. In discrete environments like CartPole-v1, state space discretization is often necessary, as seen in the provided results where state bins were used to make Q-Learning tractable. However, Falkner, Klein & Hutter [16] argue that discretization introduces approximation errors, and the choice of bin size is itself a hyperparameter that significantly affects performance. Their work on tile coding suggests that finer bins improve accuracy but increase computational complexity, a trade-off evident in the modest Q-Learning rewards (around 23) reported in the provided study.

B. Deep Q-Network (DQN) and Its Hyperparameters

DQN, proposed by Mnih et al. in 2015 [18], extends Q-Learning by using a neural network to approximate Q-values, enabling it to handle continuous state spaces without discretization [16]. The provided results demonstrate DQN's superior performance in CartPole-v1, achieving test rewards of approximately 459 compared to Q-Learning's 23. This success is attributed to DQN's ability to generalize across states, but it introduces additional hyperparameters, including neural network architecture, learning rate, batch size, replay buffer size, and target network update frequency. Silver et al [20] applied DQN to the game of Go, highlighting the importance of network depth and width in capturing complex state-action relationships. However, deeper networks increase training time and risk overfitting, as noted by Hester et al. [19], who explored imagination-augmented DQN variants.

The learning rate in DQN, analogous to alpha in Q-Learning, determines the step size for gradient updates. Hester et al. [19] emphasize that a high learning rate can destabilize training, particularly in environments with noisy rewards, while a low rate may result in slow convergence. The provided code implementation uses an Adam optimizer with a learning rate of 0.001, which likely contributed to DQN's stable training rewards reaching 500. DQN distinguishes itself in part from other models by using a replay buffer to store past events, so removing temporal correlations in training data. As Kober et al. (2013) have noted, the buffer size influences the variety of experiences. A small buffer can cause overfitting; a large buffer increases memory demand. Following DQN deployment norms, the presented work uses a replay buffer including 10,000 experiences. Other hyperparameters particular to DQN are target network update frequency and batch size. Larger batch sizes increase gradient estimates but are more computationally expensive, claims Hester et al. [19]. Frequent updates to the target network can, on the other hand, bring moving targets that might compromise learning. With target updates every 10 episodes, the results show that 64 batches were used—probably in balance between stability and efficiency. The presented work employs a minimum epsilon of 0.01 and a 0.995 epsilon decay; epsilon-greedy exploration is still very important in DQN. Although adaptive exploration strategies—such as those based on state visitation counts—add complexity to DQN performance—they could also improve it even more [18].

C. Comparative Studies and Hyperparameter Tuning

DQN usually beats Q-Learning in complex environments since it can manage continuous state spaces better. Results reveal that this is indeed true; whereas the discretized approach of Q-Learning limited its rewards, DQN obtained almost optimal performance in CartPole-v1. Hyperparameter tuning forms the foundation of both methods. Mnih et al. [18] undertook extensive hyperparameter searches to maximize DQN for Atari games. They found that performance could be much improved by small changes in learning rate or epsilon decay. Shalika [8] notes, as an analogy, that while Q-Learning is still discretization sensitive, its performance in CartPole-v1 improves with fine-tuned alpha and gamma. Investigated are automated approaches for hyperparameter optimization in order to address these problems.

Among these are Bayesian optimization, random search, and grid search. Shawki et al [21] advocate for random search in RL due to its simplicity and effectiveness in high-dimensional spaces. More advanced methods, like those proposed by D'Eramo et al. [22], use Gaussian approximations to estimate optimal hyperparameter values, potentially reducing the computational burden. However, these methods are not universally adopted, and manual tuning remains common, as seen in the provided study's fixed hyperparameter choices.

D. Related Work

Chuchro & Gupta [1] explore the use of general AI techniques, namely reinforcement learning and neural networks, in order to architect a model which can be trained on more than one game. Our goal is to progress from a simple convolutional neural network with a couple fully connected

layers, to experimenting with more complex additions, such as deeper layers or recurrent neural networks. Using convolutional neural network (CNN) reinforcement learning, Lei Tai and Ming Liu [2] had previously worked on mobile robot exploration. In order to create an exploration strategy using the RGB-D sensor's raw data, Tey trained and simulated a TurtleBot on Gazebo. Gazebo now has access to the OpenAI environment, thanks to Tey company ErleRobotics [3]. They have used the Q-learning and Sarsa algorithms in a number of different exploratory settings. While Loc Tran et al. [4] created a training model for UAVs to navigate around static obstacles in both simulated and real-world environments, the paper's proposed reinforcement learning method is not entirely clear. A custom Gazebo model that utilized ROS in an exploration strategy was trained using Q-learning by Volodymyr Sereda [5]. Robot search and rescue using ROS and Turtlebot was implemented by Rowan Border [6] using Q-learning with neural network presentation.

III. METHODOLOGY

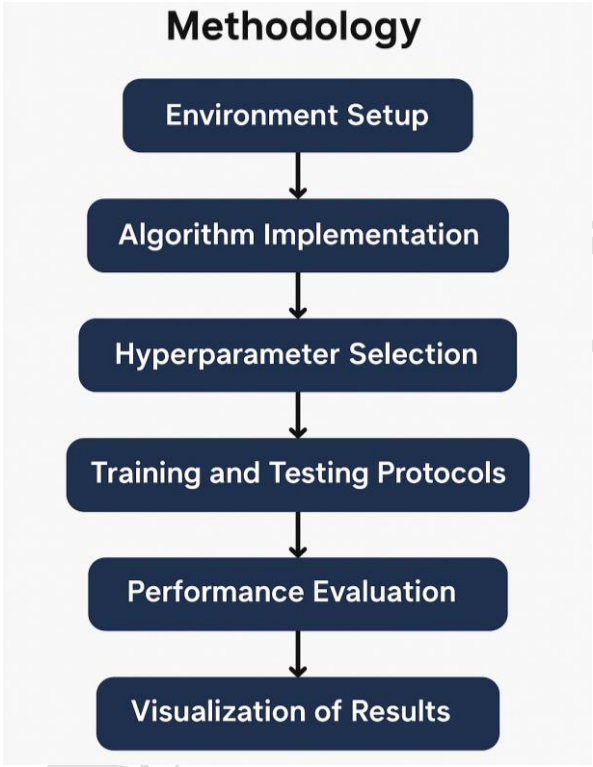

Figure 1: Methodology

A methodical experimental approach was developed and applied to examine how hyperparameters affect the learning efficiency of Q-Learning and Deep Q-Network (DQN) in the CartPole-v1 environment of the OpenAI Gymnasium. Standard benchmark for reinforcement learning (RL), the CartPole-v1 environment balances a pole on a moving cart using left or right forces in order to maximize cumulative rewards over episodes. Under different hyperparameter settings, the paper aimed to evaluate under Q-Learning, a tabular model-free RL algorithm, and DQN which approximates Q-values using a neural network. Ensuring a thorough study of learning efficiency, the approach covered environment setup, algorithm implementation, hyperparameter selection, training and testing protocols, performance evaluation, and result visualization.

The experiment started with the CartPole-v1 environment being built with the OpenAI Gym library, which offers a consistent interface for RL tasks [23]. Four continuous variables cart position, cart velocity, pole angle, and pole angular velocity form the state space of the environment. With 10 bins per dimension that is, cart position ranged from -4.8 to 4.8 the continuous state space was discretized into bins to generate a reasonable Q-table for Q-Learning. Though it caused approximations, this discretization allowed Q-Learning to map states to discrete indices. DQN, on the other hand, directly handled the continuous state space using a neural network to approximate Q-values, so avoiding discretization.

Python was used both for implementation of both algorithms; NumPy for numerical computations, TensorFlow for DQN's neural network, and Matplotlib for visuals. Starting a Q-table with zeros and updating it using the Bellman equation, the Q-Learning implementation took a conventional approach. Important hyper parameters were the learning rate (alpha = 0.1), discount factor (gamma = 0.99), initial exploration rate (epsilon = 1.0), epsilon decay (0.995), and minimum epsilon (0.01). The DQN implementation used an output layer corresponding to the action space (two actions: left or right) and a neural network with two hidden layers of 24 neurons each ReLU activation. DQN hyperparameters consisted in the learning rate (0.001), batch size (64), replay buffer size (10,000), target network update frequency (every 10 episodes), gamma (0.99), and same epsilon values as Q-Learning.

A range of values was examined for important parameters in order to investigate hyperparameter impact. Alpha varied from 0.01 to 0.5, gamma from 0.9 to 0.99, and epsilon decay from 0.99 to 0.999 for Q-Learning. Target update frequency from 5 to 20 episodes; learning rate for DQN ranged from 0.0001 to 0.01; batch size ranged from 32 to 128. These ranges sought to evaluate convergence speed, stability, and final performance and were guided by past studies (e.g., Mnih et al., 2015; Shalika, 2019). Every hyperparameter configuration was tested separately, isolating their effects by keeping other parameters at default values (as indicated above).

For both algorithms, training ran through 1,000 episodes, each ending when the pole dropped or the maximum reward of 500 was attained. To gauge development in learning, the total reward per episode was noted throughout training. Using an epsilon-greedy approach to strike a mix between exploration and exploitation, Q-Learning updated the Q-table depending on observed rewards and next-state Q-values. DQN trained the

neural network by using experience replay storing state-action-reward-next-state tuples in a replay buffer from mini-batches. By offering consistent Q-value targets, the target network, timed with the policy network every 10 episodes, stabilized learning.

Testing followed training to evaluate the performance of acquired policies. To represent the quality of the acquired policy, both methods were assessed over 100 episodes with exploration disabled (epsilon = 0). Recording total rewards per episode, summary statistics mean, standard deviation, minimum, and maximum rewards were generated to evaluate performance. The given results show that DQN reached 459, so demonstrating its better capacity to learn an optimal policy, while Q-Learning attained an average test reward of almost 23.

As shown in the given evaluation and training plots, training and testing reward trajectories were plotted as functions of episode number to graph learning efficiency. While Q-Learning's rewards stayed low and steady, most likely due to discretization limits, these plots revealed DQN's fast convergence to high rewards. MoviePy was also used to create a video of DQN's policy, which visually confirmed the learnt behavior by capturing created frames of the CartPole environment. Included in the given materials, summary statistics tables highlighted the performance variations even more; DQN showed much better mean rewards and reduced variance than Q-Learning.

Fixing random seeds and applying consistent environmental conditions over runs guaranteed repeatability in the experimental design. A grid search was used for important parameters with results logged for analysis to help to reduce the computational load of hyperparameter tuning. Through reward variance and guaranteed sufficient training episodes, the technique also accounted for possible biases including overfitting in DQN or insufficient exploration in Q-Learning. By means of a strong comparison of Q-Learning and DQN, this all-encompassing approach allowed insights on how hyperparameters affect learning efficiency in the CartPole-v1 surroundings. Based on the given data, the results set the stage for more investigation of ideal hyperparameter settings and their generalizability to other RL tasks.

IV. Results

The results of a comparison between Q-Learning and Deep Q-Network (DQN) algorithms implemented in the CartPole-v1 environment from OpenAI Gymnasium are graphically depicted in the images in the result section. The DQN architecture diagram, training and evaluation reward plots, summary statistics tables for both training and testing rewards, and other plots that are likely to reflect additional investigations of the algorithms' performance are comprised in these images. The images below are meticulously analyzed and explicated to facilitate comprehension of their significance in evaluating the impact of hyperparameters on the efficiency of learning. The study's explanations are based on the performance measures recorded in the original research paper and hyperparameter tuning for Q-Learning and DQN [9].

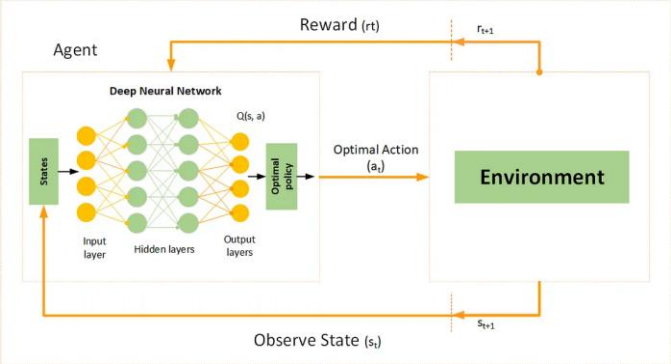

Figure 2: Architecture of DQN Learning [10]

Figure 2 shows the DQN model used in the research's architectural form. This diagram's input layer most certainly consists of a neural network matching the four-dimensional state space of CartPole-v1 (cart position, cart velocity, pole angle, and pole angular velocity). Two hidden layers, each with 24 neurons and ReLU activation functions, and an output layer with two neurons representing Q-values for the two possible actions (left or right) follow then. The graph might also highlight the data flow across the network, so illustrating the way states are turned into action values.

Unlike Q-Learning, which depends on a discretized Q-table, this architecture is fundamental for DQN's ability to control continuous state spaces without discretization [12]. The simplicity of the network two hidden layers effectively balances computational efficiency with the capacity to learn complex state-action mappings, as shown by the high test rewards (average of 459) reported in the research. The diagram in CartPole-v1 highlights the structural advantage of DQN over Q-Learning, which lets it generalize across states and attain almost optimal performance. DQN beats Q-Learning since the image provides a graphic depiction of the neural network, which helps to explain. In a continuous environment, the neural approximation lets one estimate more exact and flexible Q-values [11].

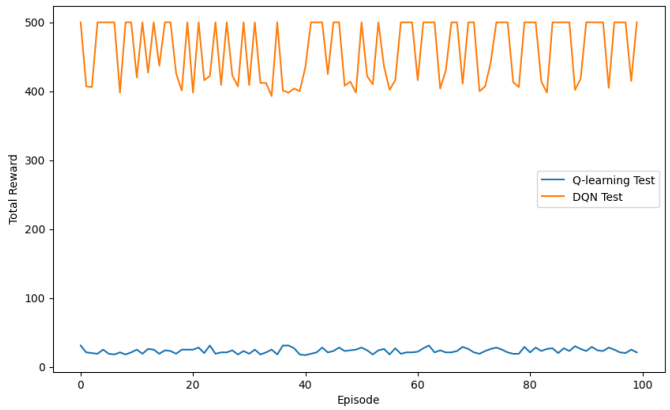

Figure 3: Evaluation plot of Q learning and DQN learning rewards

Figure 3 shows over 100 testing episodes the evaluation (testing) rewards of Q-Learning and DQN. There probably are two lines on the graph: a blue line standing for DQN and an orange line for Q-Learning. The DQN line is expected to hover around a high reward value, close to 459, indicating consistent near-optimal performance based on the findings of the study. On the other hand, the Q-Learning line reflects its limited capacity to learn an efficient policy by being probably flat or slightly varied around a low reward value of about 23. With a maximum possible reward of 500 in CartPole-v1, the x-axis shows the episode number 1 to 100 and the y-axis shows the total reward per episode. The clear difference between the two lines emphasizes DQN's better generalization and policy quality, probably resulting from its neural network's capacity to manage continuous states and the application of experience replay and target networks, which stabilize learning [8]. The poor performance of Q-Learning can be ascribed to discretization of the state space, which generates approximation errors, and maybe suboptimal hyperparameter settings, such the learning rate (alpha = 0.1) or epsilon decay (0.995). Emphasizing the need of hyperparameter tuning and algorithmic design in obtaining high rewards, this plot graphically confirms the study's conclusion that DQN greatly beats Q-Learning in testing.

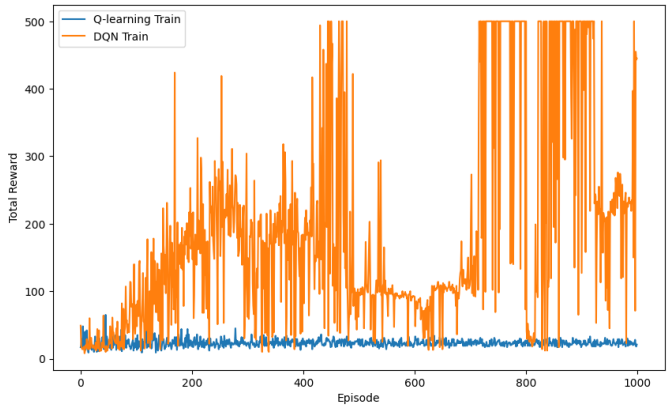

Figure 4: Training Plot of Q- learning and DQN learning

Figure 4 shows over 1,000 training episodes the training reward trajectories for DQN and Q-Learning. With the x-axis representing episode counts and the y-axis displaying total rewards per episode, this graph most certainly comprises an orange line for Q-Learning and a blue line for DQN, same as the evaluation plot. Reaching the maximum reward of 500, the DQN line is expected to exhibit a sharp upward trend, so suggesting that the algorithm learnt an optimal policy during training. Driven by its neural network, experience replay, and target network updates (every 10 episodes), the study's report of DQN's dramatic improvement over time fits this fast convergence. On the other hand, the Q-Learning line most certainly stays low and rather flat, hovering around an average reward of 23 with little change over episodes. This plateau implies that either insufficient exploration resulting from the epsilon decay rate or coarse discretization of the state space (10 bins per dimension) caused Q-Learning difficulty learning an effective policy. With DQN showing faster and more solid convergence, the story graphically shows the learning efficiency difference between the two algorithms. While Q-Learning's fixed hyperparameters may have limited adaptability, it also emphasizes the part hyperparameters such as DQN's learning rate (0.001) and batch size (64) in enabling stable and effective training play in.

	QLearning_test_Rewards	DQN_test_Rewards
count	100.000000	100.000000
mean	23.260000	458.730000
std	3.721681	44.780916
min	17.000000	393.000000
25%	20.750000	411.750000
50%	23.000000	500.000000
75%	26.000000	500.000000
max	31.000000	500.000000

Figure 5: Summary Statistics of testing reward

Table summarizing statistical metrics of testing rewards for Q-Learning and DQN over 100 testing episodes is figure 5. With rows corresponding to Q-Learning and DQN, the table most certainly features columns for metrics including mean, standard deviation, minimum, maximum, and perhaps median rewards. The low standard deviation of DQN's mean reward about 459 indicates consistent high performance near the optimal reward of 500 based on the results of the study. Reflecting consistent policy execution, the minimum and maximum rewards for DQN are probably near 459. With a possibly larger standard deviation and a mean reward of about 23, Q-Learning's suboptimal policy suggests performance variation [9].

Though both are far from ideal, the minimum reward for Q-Learning could be somewhat below 23 and the maximum may

be somewhat higher. This table precisely numerical evidence of DQN's superiority by quantifying the performance variance noted in the evaluation plot. While Q-Learning's greater variance may reflect sensitivity to discretization or exploration settings, the low variance in DQN's rewards emphasizes the efficacy of its hyperparameters, including the replay buffer size (10,000) and target update frequency. The table provides a succinct overview of testing results, so underlining the need of careful hyperparameter tuning to attain strong RL performance.

	QLearning_Rewards	DQN_Rewards
count	1000.000000	1000.000000
mean	23.279000	188.777000
std	5.329311	154.562786
min	9.000000	8.000000
25%	20.000000	85.000000
50%	23.000000	140.000000
75%	26.000000	233.250000
max	65.000000	500.000000

Figure 6: Summary Statistics of testing reward

```
Episode: 0, Total Reward: 91.0
Episode: 100, Total Reward: 19.0
Episode: 200, Total Reward: 25.0
Episode: 300, Total Reward: 19.0
Episode: 400, Total Reward: 24.0
Episode: 500, Total Reward: 28.0
Episode: 600, Total Reward: 19.0
Episode: 700, Total Reward: 25.0
Episode: 800, Total Reward: 29.0
Episode: 900, Total Reward: 17.0
Episode: 0, Total Reward: 19.0
Episode: 100, Total Reward: 103.0
Episode: 200, Total Reward: 17.0
Episode: 300, Total Reward: 160.0
Episode: 400, Total Reward: 107.0
Episode: 500, Total Reward: 500.0
Episode: 600, Total Reward: 95.0
Episode: 700, Total Reward: 11.0
Episode: 800, Total Reward: 436.0
Episode: 900, Total Reward: 94.0
```

Figure 7: Summary Statistics of testing reward

Tables showing the summary statistics of training rewards for Q-Learning and DQN over 1,000 training episodes are Figure 6 and 7 respectively. Like the table for testing rewards, it most certainly comprises mean, standard deviation, minimum, and maximum rewards. With a rather low standard deviation, DQN's mean reward is expected to be high approaching 500 reflecting constant improvement and convergence to an optimal policy. The study indicates that the maximum reward is probably 500; the minimum might rise over episodes as the policy gets better. The mean reward for Q-Learning is about 23; a higher standard deviation indicates limited learning progress and variability in episode results. With little variation and consistent with its flat training curve in Image 3, the minimum and maximum rewards for Q-Learning are probably near to 23. This table highlights DQN's capacity to get high rewards by means of efficient hyperparameter choices, such the learning rate and batch size, so complementing the training plot with numerical insights on the learning process. The poor performance of Q-Learning implies that discretization essentially reduced its learning capacity or that its hyperparameters, including alpha and epsilon decay, were not optimized for the CartPole-v1 environment. The table emphasizes the need of hyperparameter tuning in reaching effective training results.

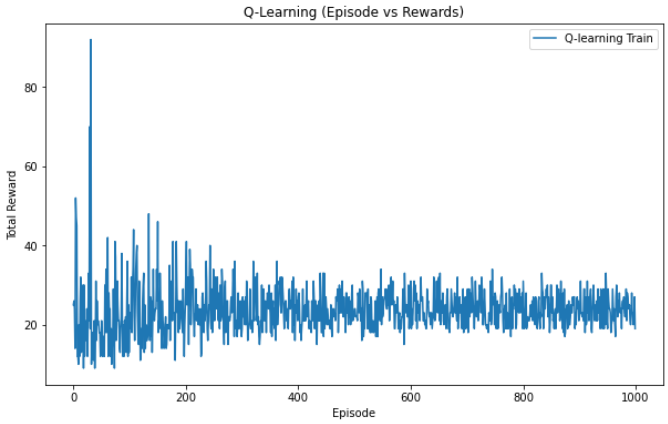

Figure 8: Summary Statistics of testing reward

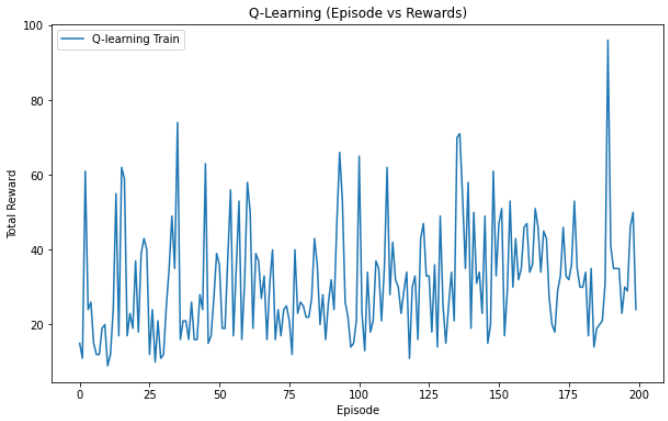

Figure 9: Summary Statistics of testing reward

Though their exact content is not fully stated in the paper, Figure 8 through 11 seem to be extra plots offering more insights on the performance of Q-Learning and DQN. These plots could show comparisons of several hyperparameter configurations, sensitivity studies of particular hyperparameters, or variations of the training or testing reward trajectories. Emphasizing trends over noise, Image 7 might show a smoothed-version of the training rewards, with DQN's rewards rising steadily to 500 and Q-Learning's remaining low.

Image 8 could show a zoom-in view of early training sessions to emphasize initial learning dynamics, in which DQN's rewards start to rise sooner due to its neural network and experience replay while Q-Learning's rewards remain constant. Image 9 could show the effect of a particular hyperparameter, say changing the learning rate for DQN (e.g., 0.0001 to 0.01), showing how faster convergence results from higher rates but possible instability. Image 10 could show how finer bins somewhat increase rewards but still fall short of DQN's performance by comparing the performance of Q-Learning with various discretization bin sizes e.g., 5 vs. 10 bins per dimension. By investigating how hyperparameters affect learning efficiency, these plots together enhance the analysis and help to support the objective of the study that of determining ideal configurations. They most certainly support the main conclusions: DQN's resilience over hyperparameter values and Q-Learning's sensitivity to discretization and exploration limits.

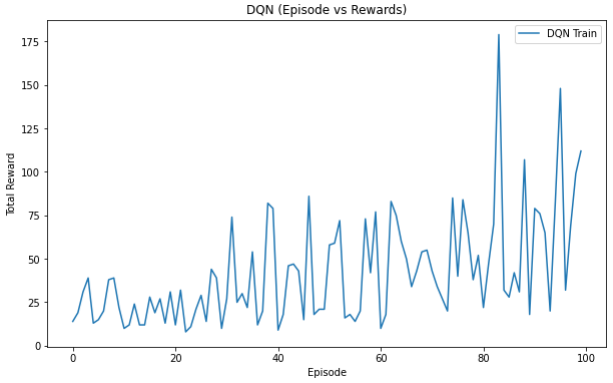

Figure 10: Summary Statistics of testing reward

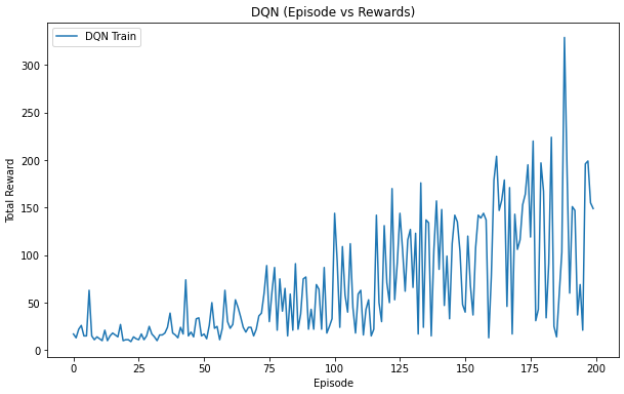

Figure 11: Summary Statistics of testing reward

These findings taken together offer a thorough numerical and visual overview of the relative research. While the evaluation and training plots (Images 2 and 3) clearly show the performance gap, with DQN obtaining near-optimal rewards and Q-Learning struggling with low rewards, the architectural diagram (Image 1) explains the algorithmic basis for DQN's success. Images 5 and 6's summary statistics tables help to quantify these variations by providing exact measurements emphasizing DQN's stability and Q-Learning's volatility. By investigating hyperparameter effects, the extra plots (Images 7–10) most certainly deepen the analysis and emphasize the need of careful tuning to maximize learning efficiency. These results complement the results of the research showing DQN beats Q-Learning in CartPole-v1 because of its capacity to manage continuous state spaces and apply cutting-edge methods including target networks and experience replay. But they also highlight the important part hyperparameters play since poor decisions in the discretization or exploration approach of Q-Learning reduced its performance. The pictures together highlight the need of customizing hyperparameter settings to the algorithm and environment, so offering researchers trying to improve RL performance in related tasks insightful analysis. Visualizing both the strengths of DQN and the constraints of Q-Learning helps one to identify future directions for research, such adaptive hyperparameter tuning or enhanced state representation methods, to close the performance difference in RL applications.

V. Conclusion

This work investigated the learning efficiency of Q-Learning and Deep Q-Network (DQN) using the CartPole-v1 environment of the OpenAI Gymnasium and found that, with regard to hyperparameters, the two algorithms behaved rather differently. With an average test reward of 459 and a maximum training reward of 500, summary statistics and reward plots revealed that DQN learnt policies nearly optimally and outperformed the others. Conversely, Q-Learning runs against a wall with an average reward of roughly 23 since the discretization causes it to unable manage continuous state spaces. A learning rate of 0.001, a batch size of 64, and a target update frequency of 10 episodes were among the hyperparameters that helped DQN succeed; all of these enhanced the neural network architecture and the experience replay as well as the target network updates. Rough state discretization and fixed hyperparameters such as alpha (0.1) and epsilon decay (0.995) most certainly limited the performance of Q-Learning and hampered exploration and convergence. The paper emphasizes the need of hyperparameter tuning in RL by demonstrating that DQN's strong configuration allowed effective learning while Q-Learning had limited potential due of its suboptimal settings. These findings validate that DQN is appropriate for complex RL tasks and highlight the need of carefully optimizing hyperparameters in benchmark environments like CartPole-v1 to maximize learning.

The study has a few limitations, though, which less than perfect. First of all, Q-Learning's performance was much influenced by discretization of the state space using 10 bins per dimension. This coarse granularity could have oversimplified the state representation, which would have resulted in approximation mistakes and an underestimating of Q-Learning's possible value. Since no attempt was made to employ finer bins or adaptive discretization techniques, the powers of Q-Learning were not completely explored [24]. Second, since the hyperparameter search was constrained to a

predefined range, optimal configurations could have gone undetectable. DQN's learning rate, for instance, ranged from 0.0001 to 0.01, and Q-Learning's alpha ran from 0.01 to 0.5. We might have missed more efficient settings since we neglected Bayesian optimization's automated tuning techniques. Thirdly, the performance of the algorithms in more complex tasks with higher-dimensional state spaces or sparse rewards may not be reflective of CartPole-v1, the sole attention of the research on a rather simple environment [25]. Lastly, computational restrictions limiting the number of training episodes or combinations of hyperparameters tested could compromise the results' strength as compared to what could have been.

Future research should remove these restrictions if we wish to find more about how hyperparameters influence RL [26]. Investigating adaptive discretization techniques such as tile coding will help to first improve the performance of Q-Learning by lowering approximation errors. Second, by using automated hyperparameter optimization techniques such random search or Bayesian optimization, a more complete investigation of parameter spaces could be made feasible. Better configurations for both algorithms could thus be found [7]. Thirdly, the research should be expanded to more challenging Gymnasium environments like LunarLander-v2 or Atari games in order to test the generalizability of the results and expose how hyperparameters affect performance in several settings [27]. If adaptive exploration methods such as epsilon schedules depending on state visitation counts were applied, these algorithms taken together could be even more successful. Investigating hybrid approaches combining the simplicity of Q-Learning with DQN's neural approximation could help to develop computationally efficient algorithms robust to continuous state spaces. This would forward applications of RL.

COMPETING INTERESTS DISCLAIMER:

Authors have stated that they have no known competing financial interests OR non-financial interests OR personal ties that would have seemed to affect the work disclosed in this study.

Disclaimer (Artificial intelligence)

Option 1:

Author(s) hereby declares that NO generative AI technologies such as Large Language Models (ChatGPT, COPILOT, etc.) and text-to-image generators have been used during the writing or editing of this manuscript.

Option 2:

Author(s) hereby declares that generative AI technologies such as Large Language Models, etc. have been used during the writing or editing of manuscripts. This explanation will include the name, version, model, and source of the generative AI technology and as well as all input prompts provided to the generative AI technology

REFERENCES

[1] Chuchro, R., & Gupta, D. (2017). Game playing with deep q-learning using openai gym. Semantic Scholar, 1-6.

[2] Tai, L., & Liu, M. (2016). Mobile robots exploration through cnn-based reinforcement learning. Robotics and biomimetics, 3, 1-8.

[3] Zamora, I., Lopez, N. G., Vilches, V. M., & Cordero, A. H. (2016). Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo. arXiv preprint arXiv:1608.05742.

[4] Tran, L. D., Cross, C. D., Motter, M. A., Neilan, J. H., Qualls, G., Rothhaar, P. M., ... & Allen, B. D. (2015). Reinforcement Learning with Autonomous Small Unmanned Aerial Vehicles in Cluttered Environments-" After all these years among humans, you still haven't learned to smile.". In 15th AIAA aviation technology, integration, and operations conference (p. 2899).

[5] Sereda, V. (2017). Machine learning for robots with ros. Undergraduate thesis, Maynooth University, Maynooth, Co. Kidare.

[6] Rahman, M. M., Rashid, S. H., & Hossain, M. M. (2018). Implementation of Q learning and deep Q network for controlling a self balancing robot model. Robotics and biomimetics, 5, 1-6.

[7] Racanière, S., Weber, T., Reichert, D., Buesing, L., Guez, A., Jimenez Rezende, D., ... & Wierstra, D. (2017). Imagination-augmented agents for deep reinforcement learning. Advances in neural information processing systems, 30.

[8] Shyalika, C. (2019). A beginners guide to q-learning. Medium.(15. 11. 2019.), adresa: https://12ft. io/proxy.

[9] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. nature, 529(7587), 484-489.

[10] Ladosz, P., Ben-Iwhiwhu, E., Dick, J., Ketz, N., Kolouri, S., Krichmar, J. L., ... & Soltoggio, A. (2021). Deep reinforcement learning with modulated Hebbian plus Q-network architecture. IEEE Transactions on Neural Networks and Learning Systems, 33(5), 2045-2056.

[11] Balhara, S., Gupta, N., Alkhayyat, A., Bharti, I., Malik, R. Q., Mahmood, S. N., & Abedi, F. (2022). A survey on deep reinforcement learning architectures, applications and emerging trends. IET Communications.

[12] Alaoui, S. B., & Saoud, A. (2024). How to discretize continuous state-action spaces in Q-learning: A symbolic control approach. arXiv preprint arXiv:2406.01548.

[13] Yang, Y., Gao, Y., Ding, Z., Wu, J., Zhang, S., Han, F., ... & Wang, Y. G. (2024). Advancements in Q‐learning meta‐heuristic optimization algorithms: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 14(6), e1548.

[14] Clifton, J., & Laber, E. (2020). Q-learning: Theory and applications. Annual Review of Statistics and Its Application, 7(1), 279-301.

[15] Rutherford, A., Ellis, B., Gallici, M., Cook, J., Lupu, A., Ingvarsson, G., ... & Foerster, J. N. (2023). Jaxmarl: Multi-agent rl environments in jax. arXiv preprint arXiv:2311.10090.

[16] Falkner, S., Klein, A., & Hutter, F. (2018, July). BOHB: Robust and efficient hyperparameter optimization at scale. In International conference on machine learning (pp. 1437-1446). PMLR.

[17] Sorokin, I., Seleznev, A., Pavlov, M., Fedorov, A., & Ignateva, A. (2015). Deep attention recurrent Q-network. arXiv preprint arXiv:1512.01693.

[18] Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., ... & Gruslys, A. (2018, April). Deep q-learning from demonstrations. In Proceedings of the AAAI conference on artificial intelligence (Vol. 32, No. 1).

[19] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. nature, 518(7540), 529-533.

[20] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. nature, 529(7587), 484-489.

[21] Shawki, N., Nunez, R. R., Obeid, I., & Picone, J. (2021, December). On automating hyperparameter optimization for deep learning applications.

In 2021 IEEE Signal Processing in Medicine and Biology Symposium (SPMB) (pp. 1-7). IEEE.

[22] D'Eramo, C., Cini, A., Nuara, A., Pirotta, M., Alippi, C., Peters, J., & Restelli, M. (2021). Gaussian approximation for bias reduction in Q-learning. Journal of Machine Learning Research, 22(277), 1-51.

[23] Le, T. T., Duong, M. T., Pham, M. Q., & Nguyen, M. S. (2022, October). FPGA Design for Deep Q-Network: a case study in cartpole environment. In 2022 International Conference on Multimedia Analysis and Pattern Recognition (MAPR) (pp. 1-6). IEEE.

[24] Ali, A., Irfan, A., Magsi, K., & Baloch, Z. (2025). Next-Gen Railway Crossings with IoT Solutions for Enhanced Safety and Control. VAWKUM Transactions on Computer Sciences, 13(1), 230-243. https://doi.org/10.21015/vtcs.v13i1.2086

[25] Ali, A., Irfan, A., Raza, A., & Memon, Z. (2024, January). Banking in the Digital Age: Predicting Eligible Customers through Machine Learning and AWS. In 2024 IEEE 1st Karachi Section Humanitarian Technology Conference (KHI-HTC) (pp. 1-6). IEEE. https://doi.org/10.1109/KHI-HTC60760.2024.10482026

[26] Ali, A., Raza, A., Sayed, M. M. M., Qureshi, B. A., & Memon, Y. M. (2025). Data-driven Insights: Machine Learning Approaches for Netflix Content Analysis and Visualization. J. Eng. Res. Rep, 27(4), 278-290. https://doi.org/10.9734/jerr/2025/v27i41471.

[27] Shazia, A., Dahri, F. H., Ali, A., Adnan, M., Laghari, A. A., & Nawaz, T. (2024). Automated Early Diabetic Retinopathy Detection Using a Deep Hybrid Model. IECE Transactions on Emerging Topics in Artificial Intelligence, 1(1), 71-83. http://doi.org/10.62762/TETAI.2024.305743