# Evaluating Deep Q-Network Variants: Insights from CartPole for Future Multi-Agent Policy Design

Bharath Muppasani
Final Project Report

***Abstract*—This project focuses on evaluating the performance of Deep $Q$-Network (DQN) variants in the CartPole environment to better understand their behavior in controlled settings. By analyzing algorithms such as Double DQN, Prioritized Replay, and Dueling Architecture, this study provides foundational insights into designing effective reinforcement learning policies. These insights can guide future efforts to devise policies for complex environments, including multi-agent systems requiring coordination and conflict avoidance.**

## I. Introduction

Reinforcement learning (RL) has made significant progress in enabling agents to learn optimal policies through interactions with their environments. A notable advancement in this field is the development of Deep Q-networks (DQNs), which combine Q-learning with deep neural networks to handle high-dimensional state spaces. However, standard DQNs can suffer from overestimation bias, leading to suboptimal policy learning. To address this, the Double DQN algorithm was introduced, decoupling action selection from evaluation by utilizing separate networks for these tasks, thereby enhancing learning stability and accuracy [1].

Another enhancement to the DQN framework is Prioritized Experience Replay, which improves learning efficiency by sampling experiences based on their temporal-difference (TD) error. This method ensures that transitions with higher learning potential are revisited more frequently, accelerating convergence and improving performance [2].

The Dueling Network Architecture further refines the DQN approach by decomposing the Q-value into separate estimations of the state value and the advantage for each action. This separation allows the learning algorithm to more effectively identify the value of different states, independent of specific actions, leading to more robust policy evaluations [3].

Evaluating these advanced DQN variants in controlled environments, such as the CartPole task, provides critical insights into their performance and potential applications. The CartPole environment serves as a benchmark for testing RL algorithms due to its simplicity and the clear objective of balancing a pole on a cart. By analyzing how these algorithms perform in such settings, we can better understand their strengths and limitations, informing future applications in more complex scenarios, including multi-agent systems where coordination and conflict avoidance are essential.

This project focuses on implementing and evaluating Double DQN, Prioritized Experience Replay, and Dueling Network Architectures within the CartPole environment. The findings aim to establish a foundational understanding of these algorithms, guiding the development of effective reinforcement learning policies for complex environments in future research.

## II. Methodology

### A. Problem Motivation

In multi-agent systems, achieving coordinated behaviors presents challenges such as conflict avoidance and effective resource allocation. While this project does not directly implement multi-agent environments, the results from single-agent experimentation serve as a stepping stone. By evaluating the performance of advanced RL frameworks, we aim to establish a basis for extending these algorithms to multi-agent setups in the future.

### B. Methodology

To evaluate the performance of DQN variants, we implemented the following algorithms in the CartPole environment:

- **Double DQN**: Mitigates overestimation bias by decoupling action selection and evaluation.
- **Prioritized Experience Replay**: Enhances learning by prioritizing experiences with higher temporal-difference errors.
- **Dueling Network Architecture**: Separates state-value and advantage components to improve policy optimization.

The CartPole environment provides a controlled benchmark to validate the functionality and efficiency of these variants.

*1) Double DQN:* The Double DQN [1] algorithm reduces overestimation bias by decoupling action selection from action evaluation. The target update for Double DQN is:

$$y_i = r + \gamma Q_{\theta^-}(s', \arg\max_a Q_\theta(s', a)), \tag{1}$$

where $s'$ is the next state, $\gamma$ is the discount factor, and $\theta$ and $\theta^-$ are the policy and target network parameters, respectively.

*2) Prioritized Experience Replay:* Prioritized Experience Replay [2] allocates memory capacity to experiences with higher TD errors. The probability of sampling an experience $i$ is defined as:

$$P(i) = \frac{\delta_i^\omega}{\sum_k \delta_k^\omega}, \tag{2}$$

where $\delta_i$ is the TD error, and $\omega$ controls prioritization intensity. This ensures critical transitions are emphasized during training.

*3) Dueling Network Architecture:* The Dueling Network Architecture [3] decomposes Q-values into state-value and advantage components, allowing for better differentiation between actions. The Q-value is approximated as:

$$Q(s,a;\theta) = V(s;\theta) + \left( A(s,a;\theta) - \frac{1}{|A|} \sum_{a'} A(s,a';\theta) \right), \tag{3}$$

where $V(s;\theta)$ is the state-value function, and $A(s,a;\theta)$ represents the advantage function.

## III. Experiment Setup

The following section details the experimental setup, including the description of the CartPole-v1 environment, the neural network architectures used, and the hyperparameters configured for the experiments. All implementation details and code used for this study are available in the GitHub repository [5].

### A. CartPole-v1 Environment

The CartPole-v1 environment from OpenAI Gym [4] was utilized as the experimental testbed. This environment presents a classic control problem where the objective is to balance a pole on a moving cart by applying forces to the cart. The agent receives a reward for each time step the pole remains balanced, with the episode terminating when the pole falls over or the cart moves out of bounds. This setup provides a straightforward yet challenging scenario for evaluating reinforcement learning algorithms.

### B. Neural Network Architectures

Two neural network architectures were implemented to approximate the Q-value function: the standard DQN and the Dueling DQN. Both architectures were constructed using PyTorch, with the following configurations:

*1) Deep Q-Network (DQN):* The DQN architecture comprises three fully connected layers:

- **Input Layer**: Accepts the state representation with dimensions corresponding to the environment's state space.
- **Hidden Layers**: Two hidden layers, each with 128 neurons and ReLU activation functions.
- **Output Layer**: Outputs Q-values for each possible action in the action space.

Weight initialization was performed using Xavier uniform initialization to facilitate efficient training convergence.

*2) Dueling Deep Q-Network (Dueling DQN):* The Dueling DQN architecture separates the estimation of the state-value and advantage functions:

- **Shared Feature Layers**:
  - Input layer processing the state representation.
  - Two hidden layers with 128 neurons each and ReLU activations.
- **Value Stream**:
  - A fully connected layer with 128 neurons and ReLU activation.
  - An output layer producing a single scalar value representing the state-value function.
- **Advantage Stream**:
  - A fully connected layer with 128 neurons and ReLU activation.
  - An output layer producing values corresponding to each action, representing the advantage function.

The Q-values are computed by combining the state-value and advantage functions, with the advantage mean subtracted to ensure stability and identifiability, as described in the dueling network architecture literature [3].

### C. Hyperparameters

The following hyperparameters were employed across all experiments to ensure consistency and facilitate comparative analysis:

- **Episodes**: 300
- **Batch Size**: 64
- **Epsilon Minimum** ($\epsilon_{\text{min}}$): 0.001
- **Epsilon Decay Rate**: 0.995
- **Discount Factor** ($\gamma$): 0.99
- **Replay Buffer Size**: $10^6$

These hyperparameters were selected based on standard practices in deep Q-learning to balance exploration and exploitation effectively.

### D. Training Procedure

The training process involved the following steps:

1) **Experience Replay**: Agents interacted with the environment, and their experiences were stored in a replay buffer of size $10^6$. Mini-batches of 64 experiences were sampled uniformly for training.
2) **Epsilon-Greedy Policy**: Action selection was governed by an epsilon-greedy policy, with epsilon decaying from an initial value to a minimum of 0.001 at a rate of 0.995 per episode, balancing exploration and exploitation.
3) **Optimization**: The network parameters were updated using the Adam optimizer.
4) **Target Network**: A separate target network was employed, with its parameters periodically updated, for every 200 training steps, to stabilize training, following the approach outlined in [1].

## IV. Results and Discussion

The following section evaluates the performance of various DQN variants implemented in the CartPole-v1 environment. Results are analyzed across two key settings: normalized reward and non-normalized reward. Each setting includes comparisons for training performance (average rewards and training loss) and Q-values analysis, highlighting specific trends and observations.

### A. Performance Overview for All Variants

We evaluated five variants of the DQN algorithm. Each variant's average reward and training loss are analyzed below:

*1) DQN:* The baseline DQN exhibited slow convergence, achieving modest rewards. The average reward increased steadily but plateaued after 150 episodes, reflecting limited policy improvement. Training loss consistently increased over time and started to converge at the end of the training process, indicating a slower learning process and challenges in stabilizing the policy.


Fig. 1: DQN: Average Reward and Training Loss.

*2) DDQN:* Double DQN mitigated overestimation bias, leading to improved reward stability compared to the baseline. The average reward demonstrated gradual improvement, slowly converging smoothly after 220 episodes. Training loss also converged more effectively, showcasing the benefits of decoupling action selection and evaluation.


Fig. 2: DDQN: Average Reward and Training Loss.

*3) DDQN + PER:* Incorporating Prioritized Experience Replay accelerated convergence, as seen from the steep increase in average reward within the first 100 episodes and a steady increase after 150 episodes. Training loss remained consistently low throughout the episodes due to the efficient sampling of critical experiences, ensuring faster and more effective learning.

*4) DDQN + Dueling:* The dueling architecture enhanced state-value estimation, leading to improved reward stability from 150 to 250 episodes. While the average reward converged at a higher level than DDQN, the training loss consistently reduced over time, highlighting the architecture's ability to


Fig. 3: DDQN + PER: Average Reward and Training Loss.

efficiently separate state values and action advantages. However, a noticeable drop in average reward was observed around 260 episodes, which may be attributed to the fixed learning rate. A dynamically annealing learning rate could potentially mitigate this issue by adapting to the learning progress and ensuring smoother convergence.

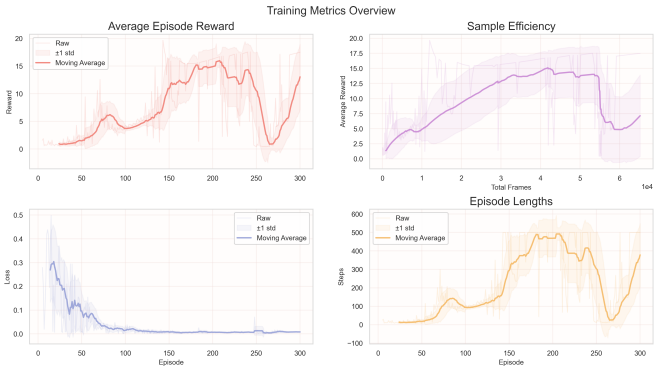



Fig. 4: DDQN + Dueling: Average Reward and Training Loss.

*5) DDQN + PER + Dueling:* Combining PER and Dueling achieved the best overall performance. The average reward increased rapidly within the first 150 episodes. Training loss remained low and stable, highlighting the complementary benefits of prioritized sampling and state-value decomposition.

### *B. Training Comparison: Normalized vs. Non-Normalized Reward*

The training performance was evaluated under both normalized and non-normalized reward settings. While normalized rewards scale the outputs for stability, non-normalized rewards retain raw values, which provide meaningful feedback in environments like CartPole, where the task has an inherent finite reward structure.

Figure 6 compares the performance of all variants across normalized and non-normalized reward settings. The following observations were made:

- In the normalized reward setting, training loss converged smoothly across all variants, correlating well with reward stability.

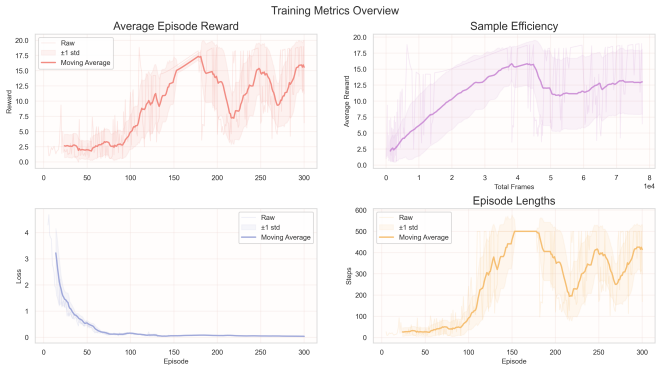



Fig. 5: DDQN + PER + Dueling: Average Reward and Training Loss.

- In the non-normalized reward setting, models incorporating PER and Dueling (DDQN + PER, DDQN + Dueling) achieved higher rewards after 200 episodes. However, this was accompanied by an increase in training loss, suggesting that larger Q-value updates are required to handle higher variance in rewards.

### C. Q-Values Analysis

We analyzed the evolution of Q-values for all DQN variants, focusing on target Q-values, current Q-values, next Q-values, and temporal-difference (TD) errors. Figure 7 illustrates these trends in the normalized reward setting:

- **Target Q-Values**: Values steadily increased across episodes for all variants, reflecting improved policy learning. DDQN + PER and DDQN + PER + Dueling achieved higher Q-value estimates earlier due to prioritized sampling.
- **Current Q-Values**: The current Q-values aligned with target values over time, stabilizing after 200 episodes. Dueling variants demonstrated smoother Q-value updates.
- **Next Q-Values**: Next Q-values gradually approached expected returns, stabilizing within 150 episodes for DDQN + PER and DDQN + PER + Dueling, which exhibited faster convergence.
- **TD Errors**: TD errors reduced consistently across all variants, with minor fluctuations observed after 200 episodes, particularly in PER-based methods, due to prioritized updates of critical transitions.

### D. Key Insights

- Non-normalized rewards remain useful in CartPole as they retain the meaningful scale of the task. Higher reward variance correlates with model performance improvement.
- The combination of DDQN with Prioritized Experience Replay and Dueling Network Architecture consistently outperformed other variants across all metrics.
- Training loss increased in non-normalized settings when the models achieved higher rewards, suggesting that larger Q-value updates lead to higher TD errors.

These results validate the effectiveness of advanced DQN variants, demonstrating their ability to optimize performance in the CartPole-v1 environment under varying reward structures. The non-normalized setting, while noisier, better reflects the natural scale of the problem, making it relevant for practical reinforcement learning scenarios.

## V. Conclusion

This study provided foundational insights through experiments on the CartPole-v1 environment, evaluating variations of Deep Q-Networks. Building on these findings, future efforts will extend these algorithms to multi-agent environments. This progression will involve centralized planning mechanisms and strategies to address inter-agent interactions, for tackling more complex reinforcement learning challenges.

## References

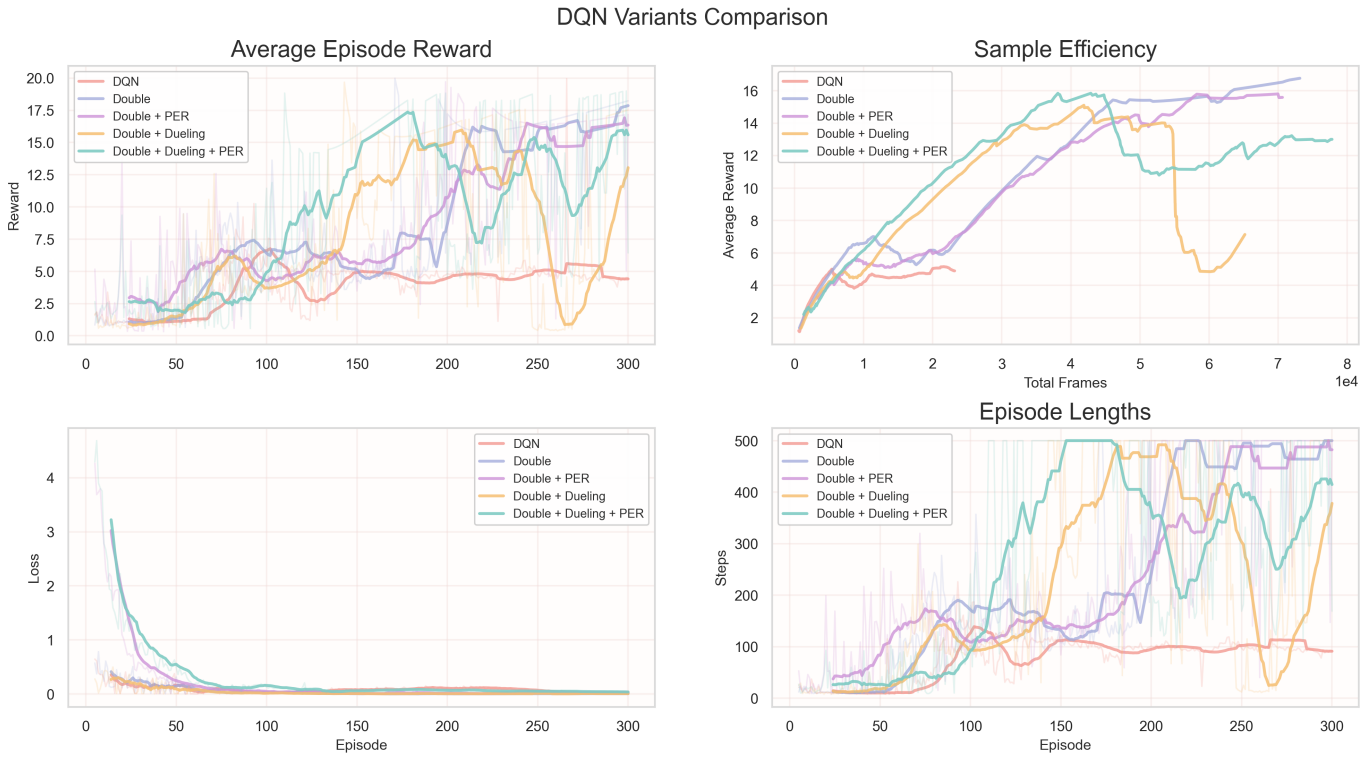
[1] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[2] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *International Conference on Learning Representations (ICLR)*, 2016.

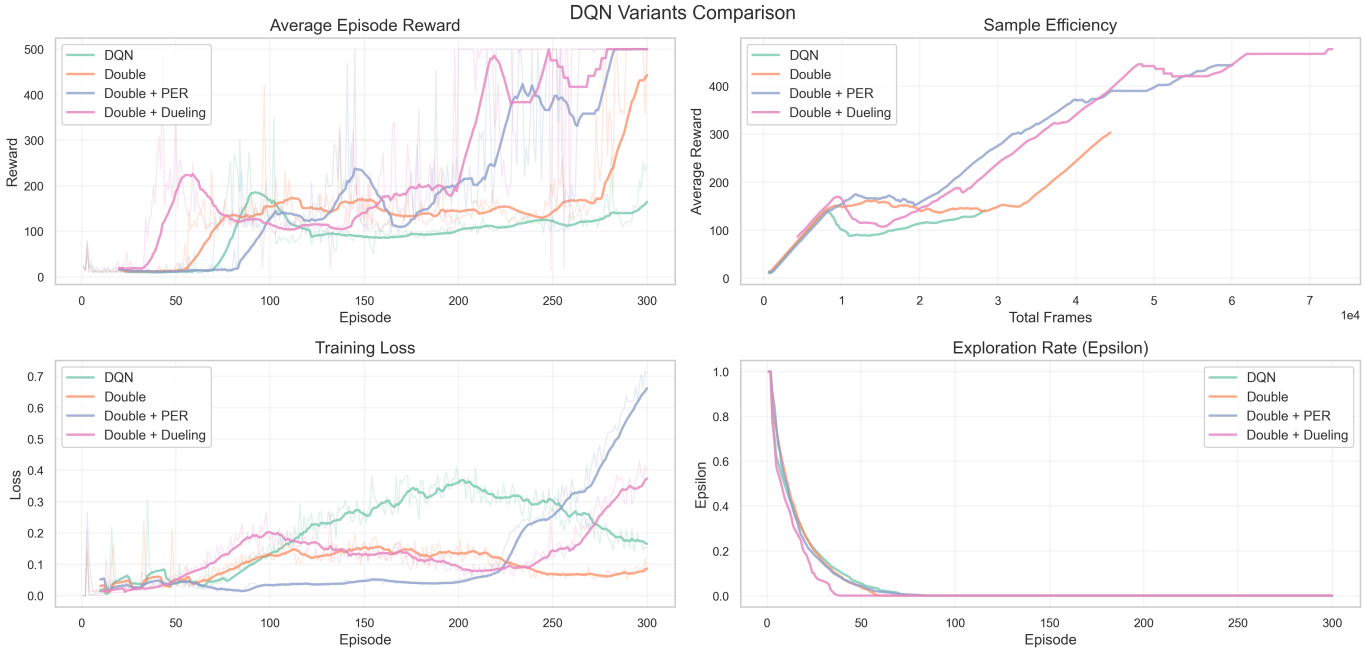
[3] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling Network Architectures for Deep Reinforcement Learning," in *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

[4] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv preprint arXiv:1606.01540*, 2016.

[5] B. Muppasani, "NN Class Project Repository," GitHub repository, 2024. [Online]. Available: https://github.com/BharathMuppasani/nn_class.




(a) Normalized Reward: Training Comparison




(b) Non-Normalized Reward: Training Comparison

Fig. 6: Training Comparison Across Normalized and Non-Normalized Rewards.


Fig. 7: Q-Values Analysis with Normalized Rewards.