Analysis Report

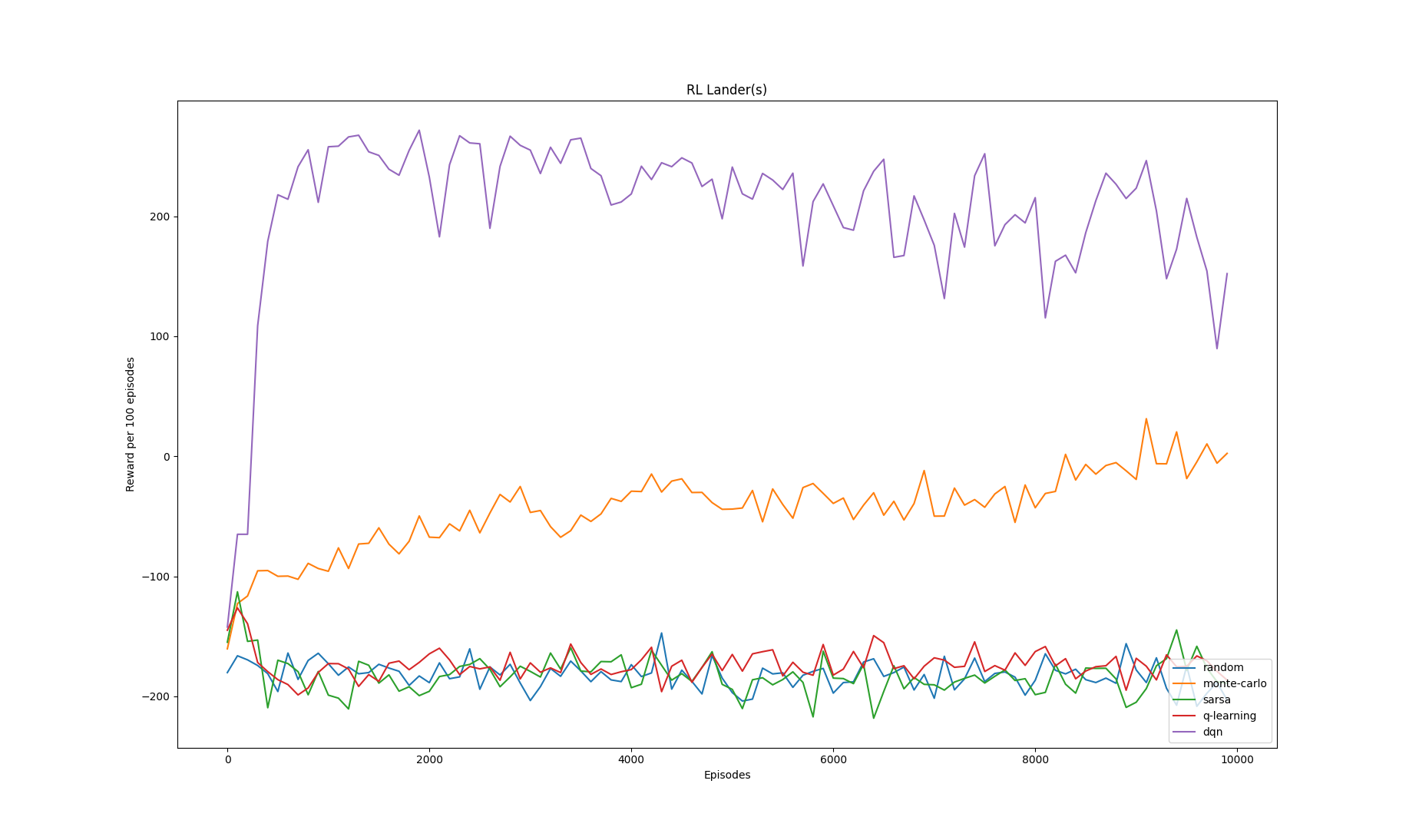
Betty Han 301202325 & Shibo Wang 301200419

Discuss the rationale of the activation functions & the loss function used in the network.

* ReLU is used for the hidden layers. It is a non-linear activation function that helps the network learn complex patterns.
* Linear Activation is used in the output layer because continuous values for each action is predicted. It is appropriate because it doesn’t restrict the range of values.
* Loss Function (Mean Squared Error - MSE): It measures the average squared difference between the predicted Q-values and the target Q-values. MSE is suitable for regression tasks.

Define the hyperparameters:

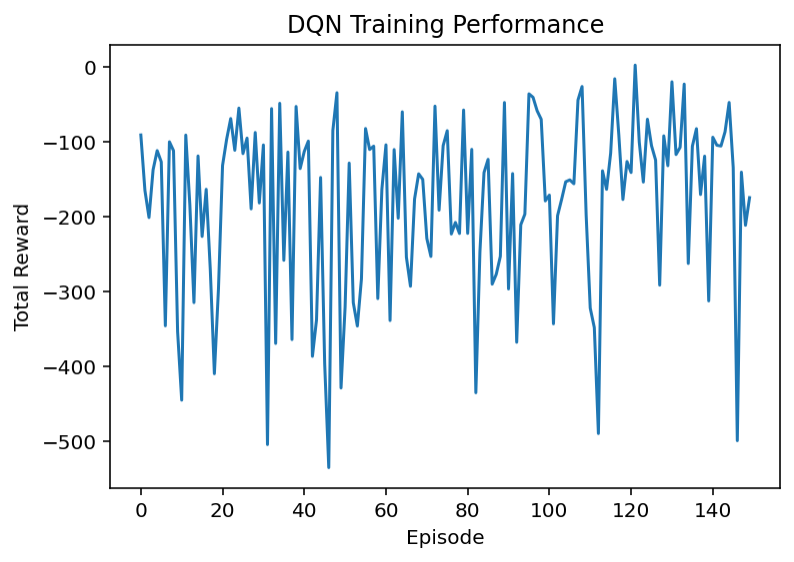
We chose the hyperparameters based on the OpenAi Gymenvironment author’s introduction: https://github.com/lazavgeridis/LunarLander-v2/tree/main, and set the best DNQ and learning rate.



* the number of iterations: 1000 \* 200 = 200,000 #num\_episodes \* max\_steps
* num\_episodes = 1000 # Number of episodes to run before updating the model
* max\_steps = 200 # Maximum steps per episode
* Discount Factor (Gamma): gamma = 0.99 # Discount factor

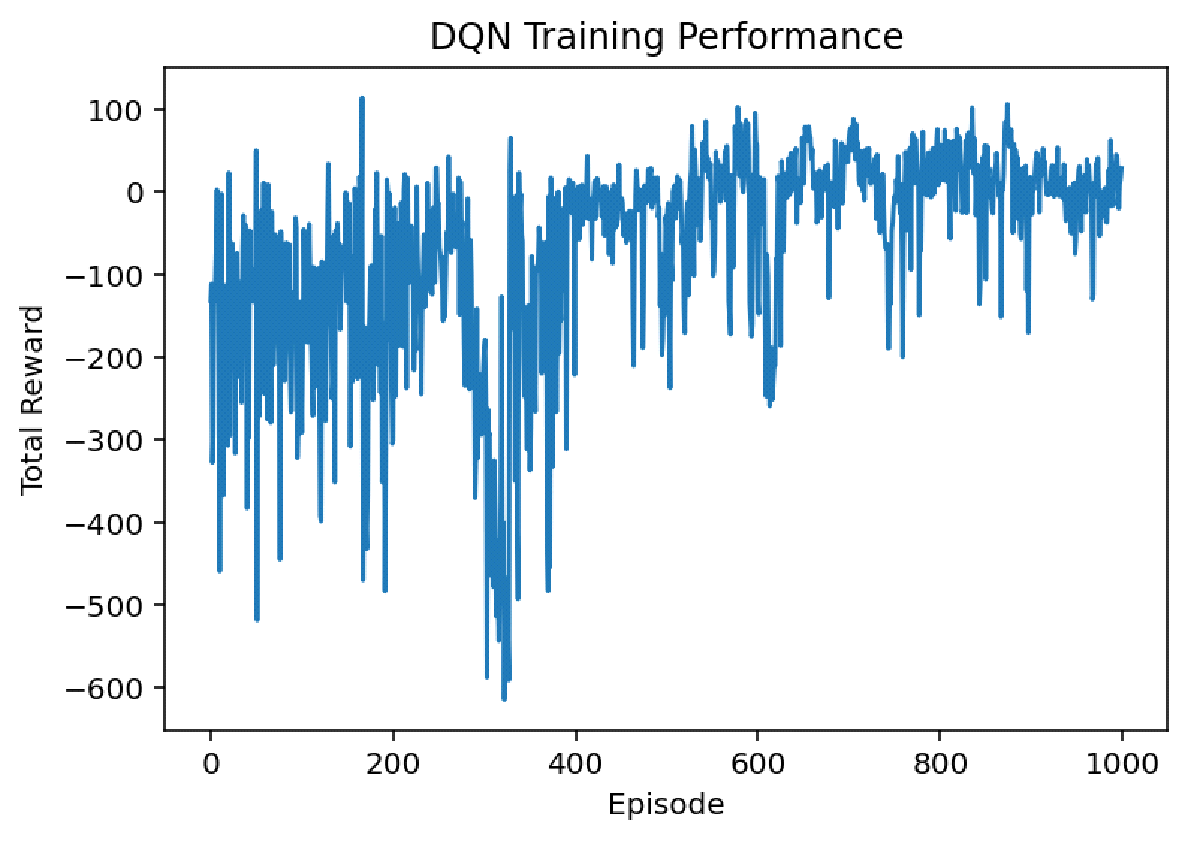
Agent's learning progress plot:

150 episodes:



We don't have one positive reward, so we increase the training episode to 1000.

1000 episodes:



Total Reward per Episode: The plot of total reward over 1000 episodes shows the agent’s learning progress. Initial fluctuations indicate exploration, while later stabilization suggests learning and improvement. Total Reward over: 15.189614490301723 @800 episodes

Discuss the challenges faced during training and potential strategies for further improving the agent's performance.

* Challenges:
* Choosing the Right Hyperparameters - The performance of the DQN agent is highly sensitive to hyperparameters such as the learning rate, batch size, gamma (discount factor), and epsilon decay rate. Poor choices for these hyperparameters can lead to suboptimal performance, slower convergence, or instability.
* Training Time: DQN agents often require a significant amount of training time and computational resources, particularly in more complex environments like LunarLander-v2. Training a model for 1000+ episodes can take several hours to days depending on the hardware.
* Potential strategies:
* Systematic Hyperparameter Tuning: Use grid search, random search, or Bayesian optimization to systematically explore the hyperparameter space.
* If there is a pre-trained agent from another environment or task, fine-tuning it on LunarLander-v2 can significantly reduce training time.