Blackjack RL Project

Kormishenkov Alexander, Ozerova Daria, Michael Kuznetsov

23.12.2024

Blackjack RL Project : Technical Specifications

Environment Rules
Deck Composition:
Infinite deck with card sampling with replacement.
Card values range from 1 to 10, uniformly distributed.
Card colors:

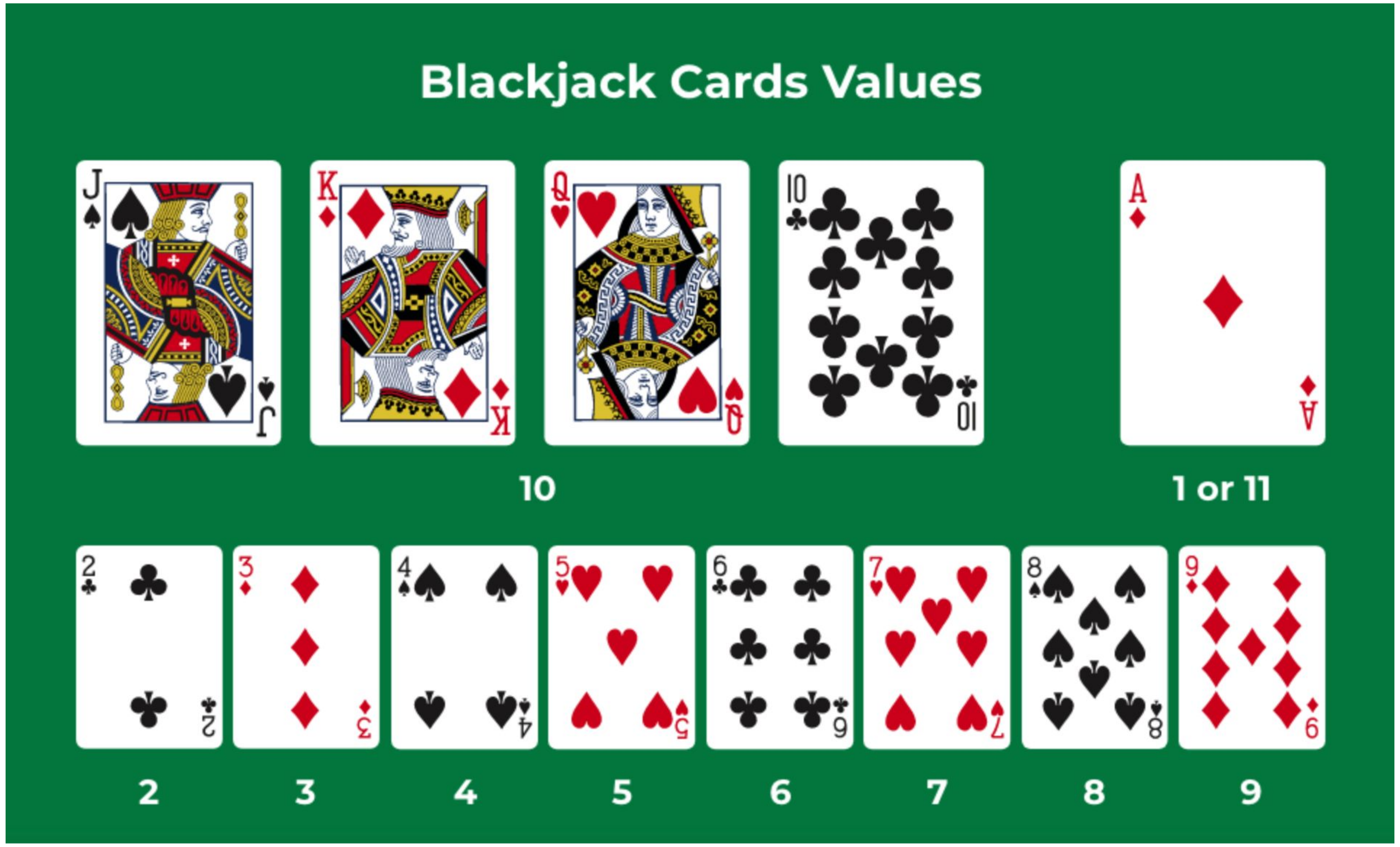
- Red (1/3 probability): Value is subtracted.
- Black (2/3 probability): Value is added.

Game Mechanics:
Both player and dealer start with one black card.
Player Actions:

- Hit: Draw another card.
- Stick: End turn with the current score.

Card Value Rules:
Player sum > 21 or < 1 → Bust (-1 reward).
Dealer follows the same rules as the player but always sticks at 17+.
Game Outcomes:

- Dealer Bust: Player wins (+1 reward).
- Player Bust: Player loses (-1 reward).
- Neither Busts: Largest valid sum wins; tie results in 0 reward.



Blackjack RL Project : Technical Specifications

Comparison Metrics

- Training Stability:
  Convergence speed and robustness of learned strategies.
- Sample Efficiency:
  Number of episodes required to reach optimal performance.
- Policy Behavior:
  Examine heatmaps of action selection to compare strategies.
  Assess the alignment with human Blackjack strategies.
- Cumulative Rewards:
  Average reward over evaluation episodes.

Deliverables

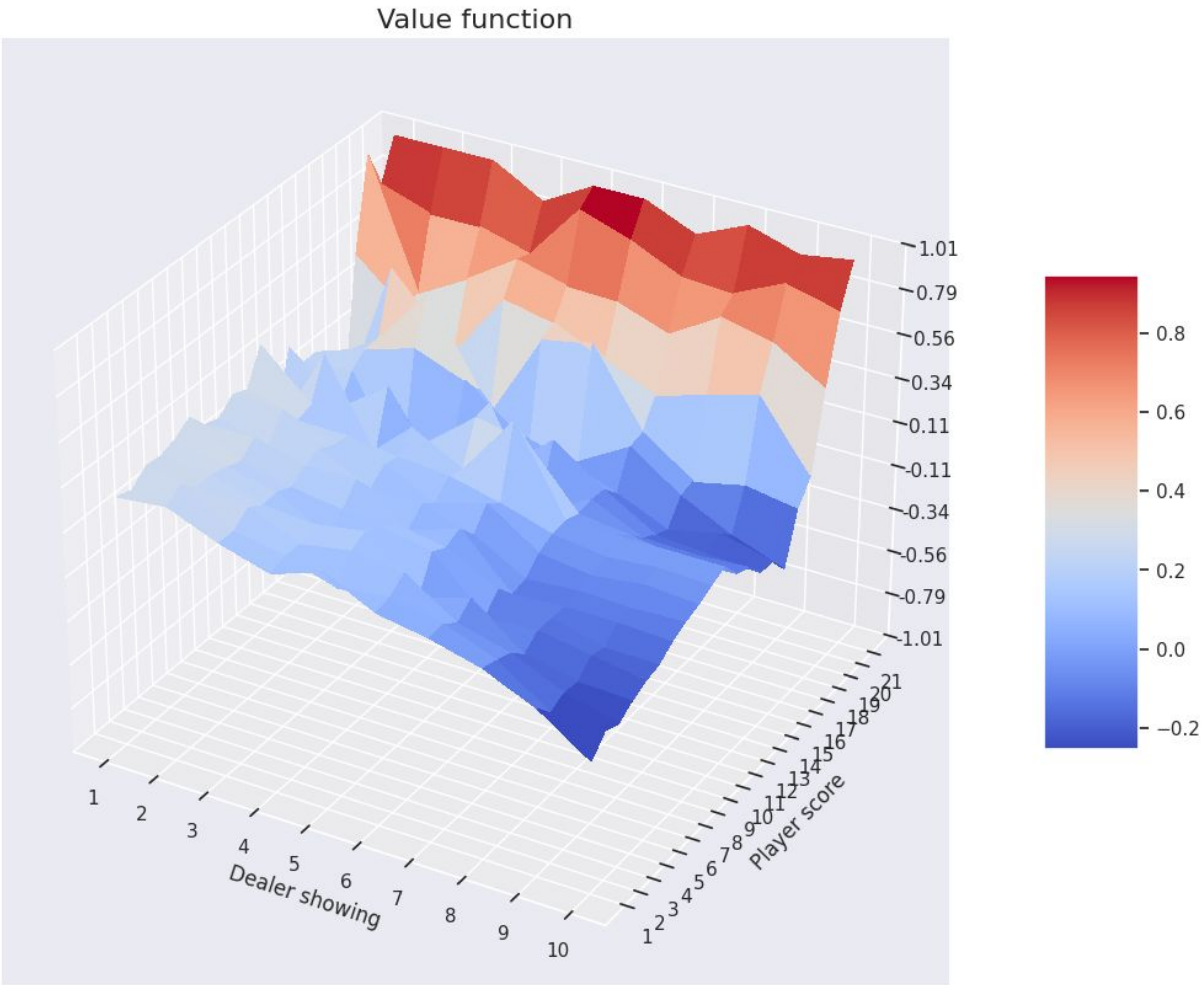
- Agents:
  PPO implementation using PyTorch.
  Monte Carlo agent with tabular Q-learning.
  DQN + DQN PER
- Visualization:
  Policy heatmaps for agents.
  Reward progression during training.
- Comparison Summary:
  Insights into the strengths and weaknesses of both agents.

Blackjack RL Project : Monte Carlo agent (MC)

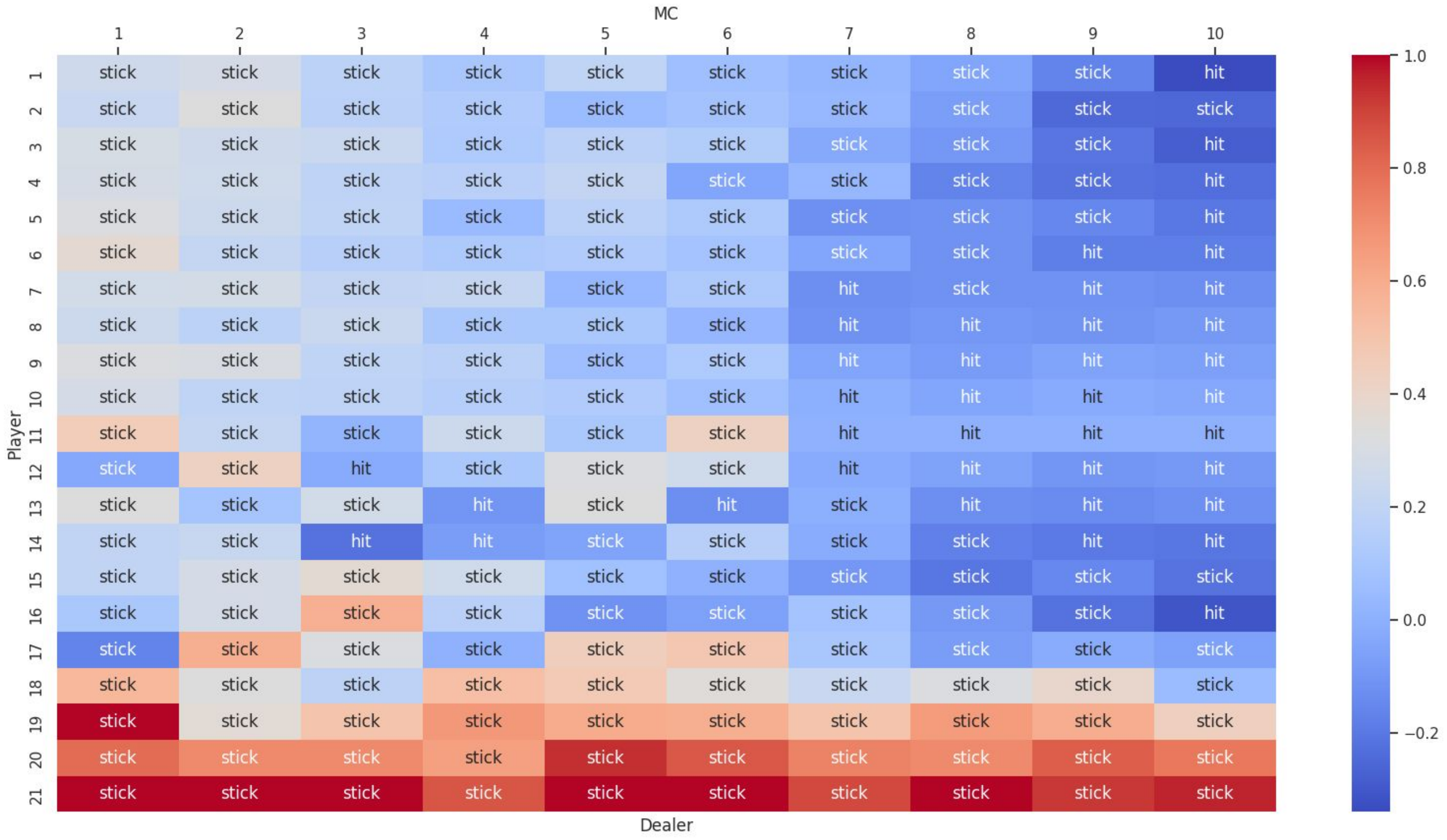
Exploration and Exploitation Policies:
- Random Policy (random_policy): Selects actions randomly.
- Epsilon-Greedy Policy (e_greedy_policy): Balances exploration and exploitation by choosing random actions with probability ϵ or the action with the highest Q-value otherwise.
- Prescribed Policy (get_action): Executes the current policy (random or epsilon-greedy) to choose an action.

State and Action Tracking:
- Counters: Tracks how often each state or state-action pair has been visited to decay the learning rate and influence exploration.
- Best Action (get_action_w_max_value): Identifies the action with the maximum Q-value for a given state.

Optimal Policy Extraction (optimal_policy):
- Constructs the optimal policy and value function from the learned Q-table.
- Produces a table summarizing the best action for each state.



Blackjack RL Project : Monte Carlo agent (MC)



Blackjack RL Project : Proximal Policy Optimization (PPO) agent

Actor-Critic Network:

Memory:

- Temporarily stores episode data including: States, actions, rewards, log probabilities, loss and terminal flags.
- Supports efficient computation of discounted rewards and advantages.

Loss Function:

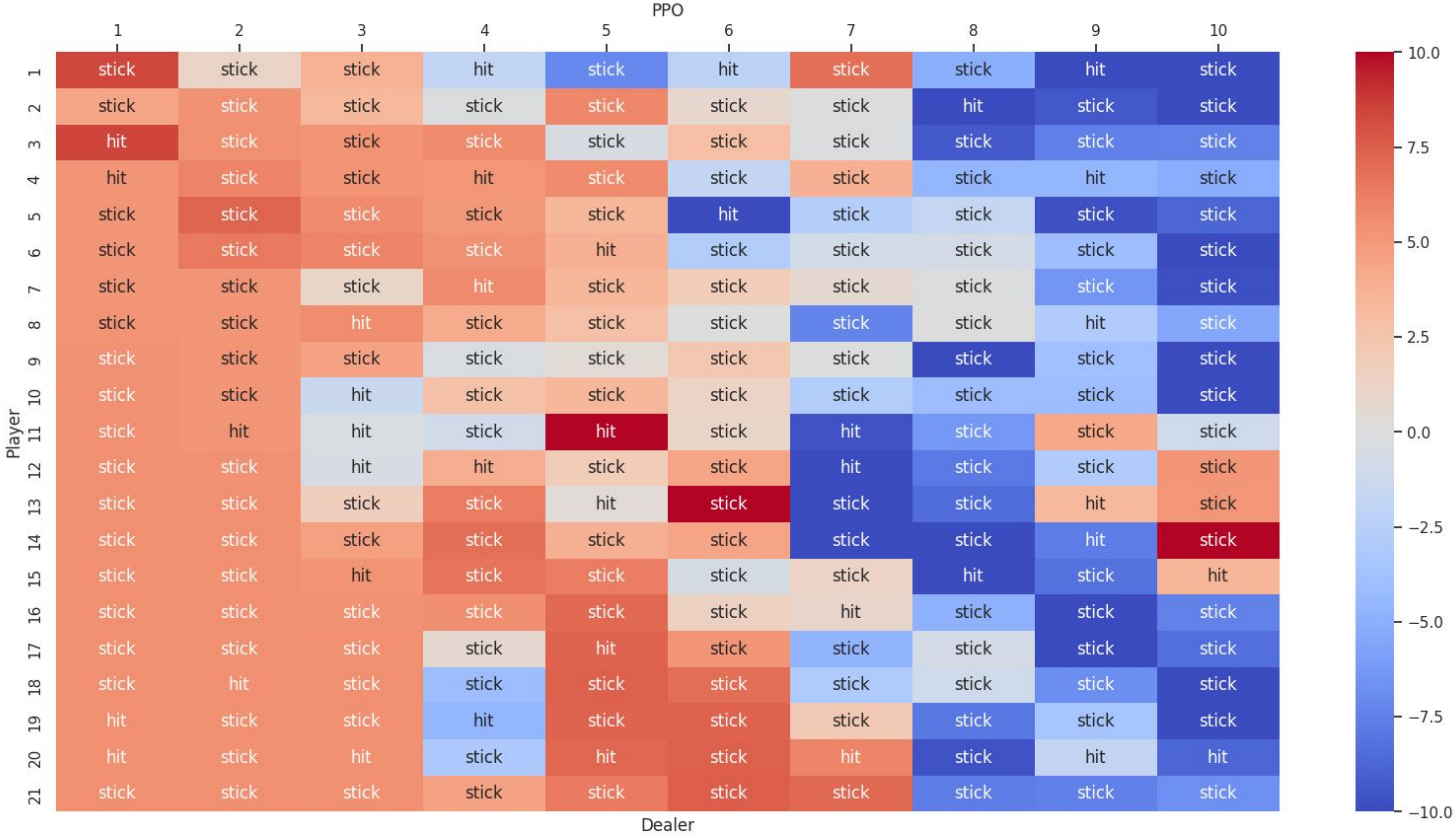
Combines two objectives:

- Clipped Surrogate Objective: Ensures that policy updates remain within a trust region to avoid large policy shifts.
- Value Function Loss: Minimizes the error in the state-value predictions.

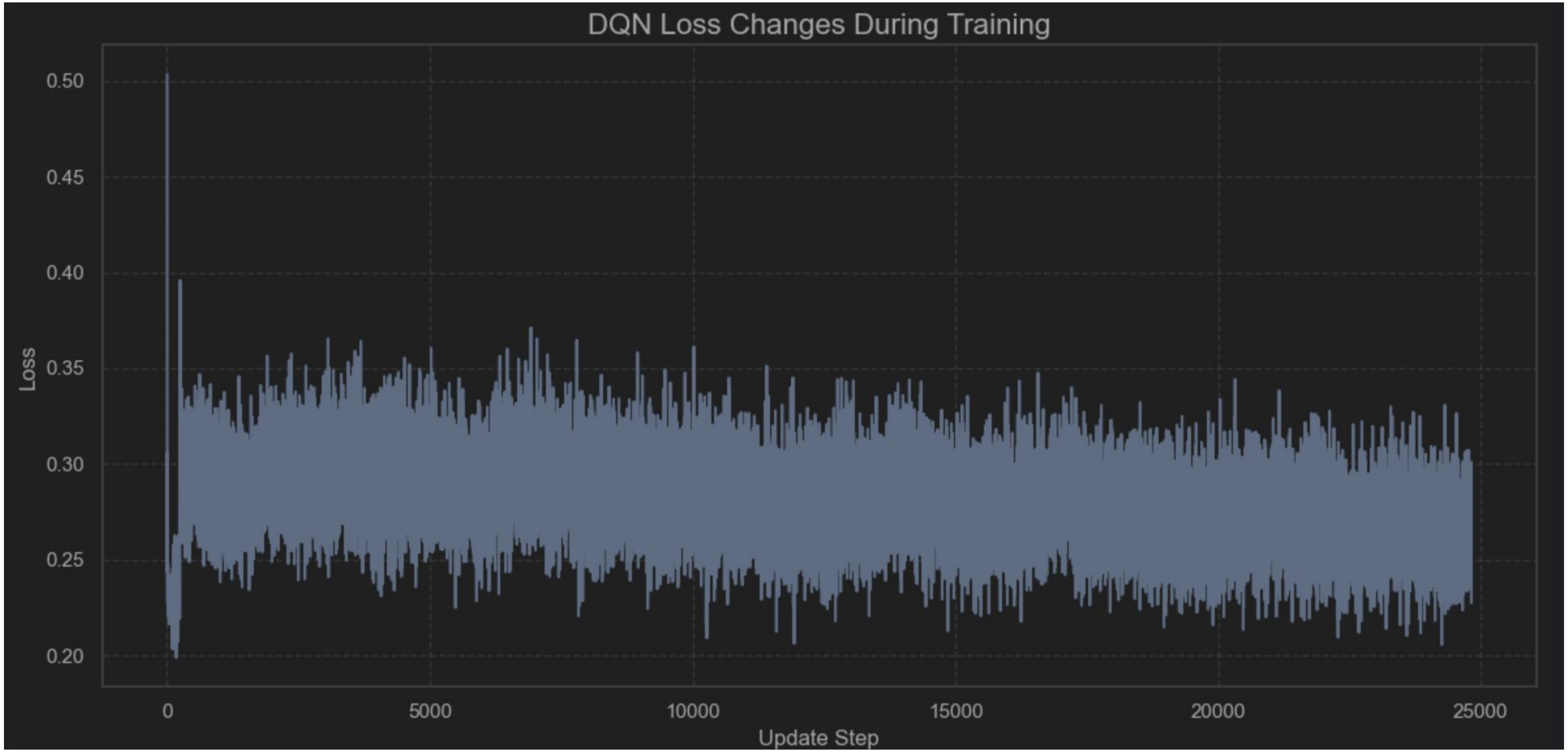
Discounted Rewards:

- Uses a discount factor (γ) to compute future rewards, balancing immediate and long-term gains.

Blackjack RL Project : Proximal Policy Optimization (PPO) agent

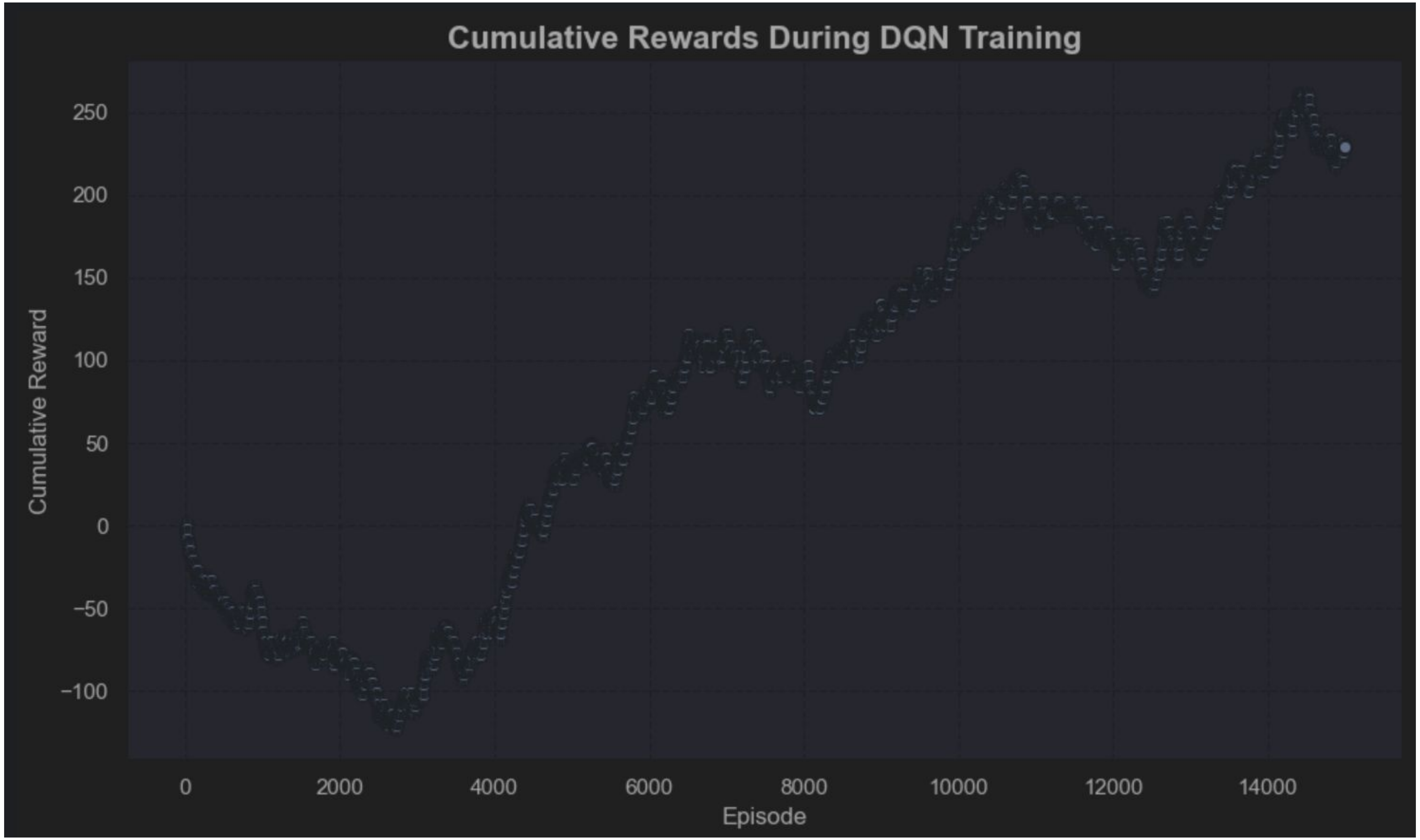


DQN



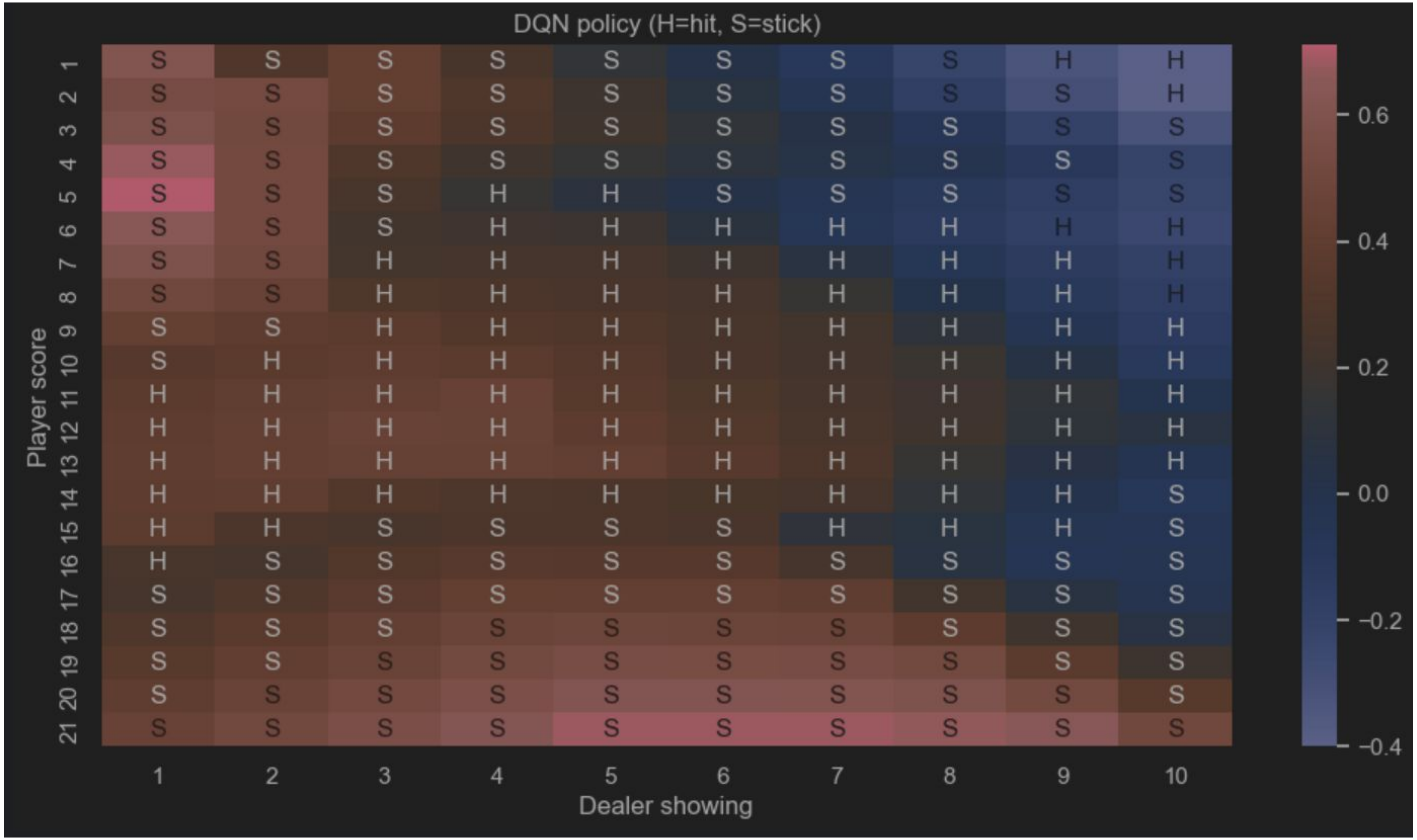
DQN Loss Changes During Training
0.50
0.45
0.40
0.35
0.30
0.25
0.20
Loss
0
5000
10000
15000
20000
25000
Update Step

DQN

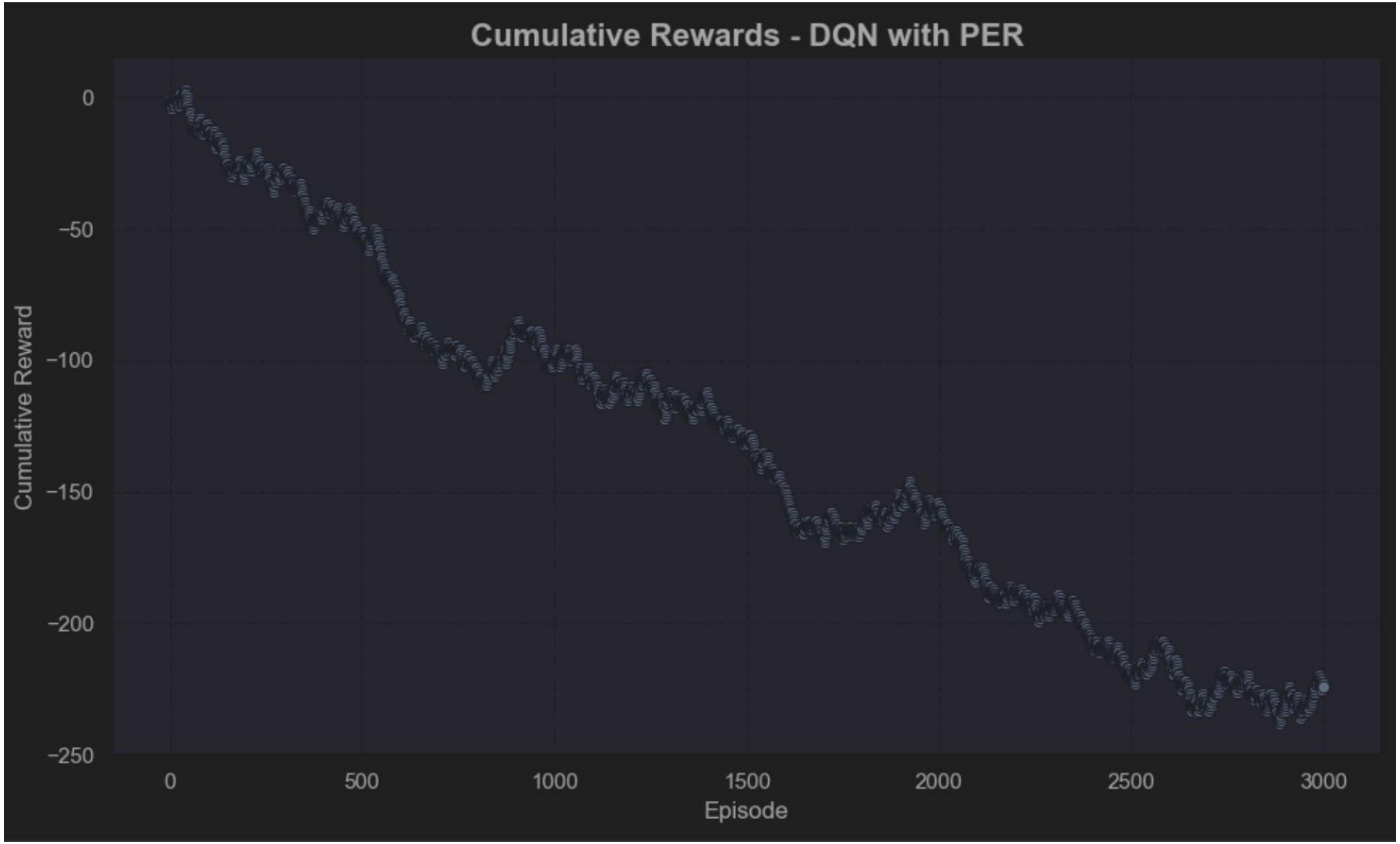


Cumulative Rewards During DQN Training
Cumulative Reward
Episode
250
200
150
100
50
0
−50
−100
0
2000
4000
6000
8000
10000
12000
14000

DQN

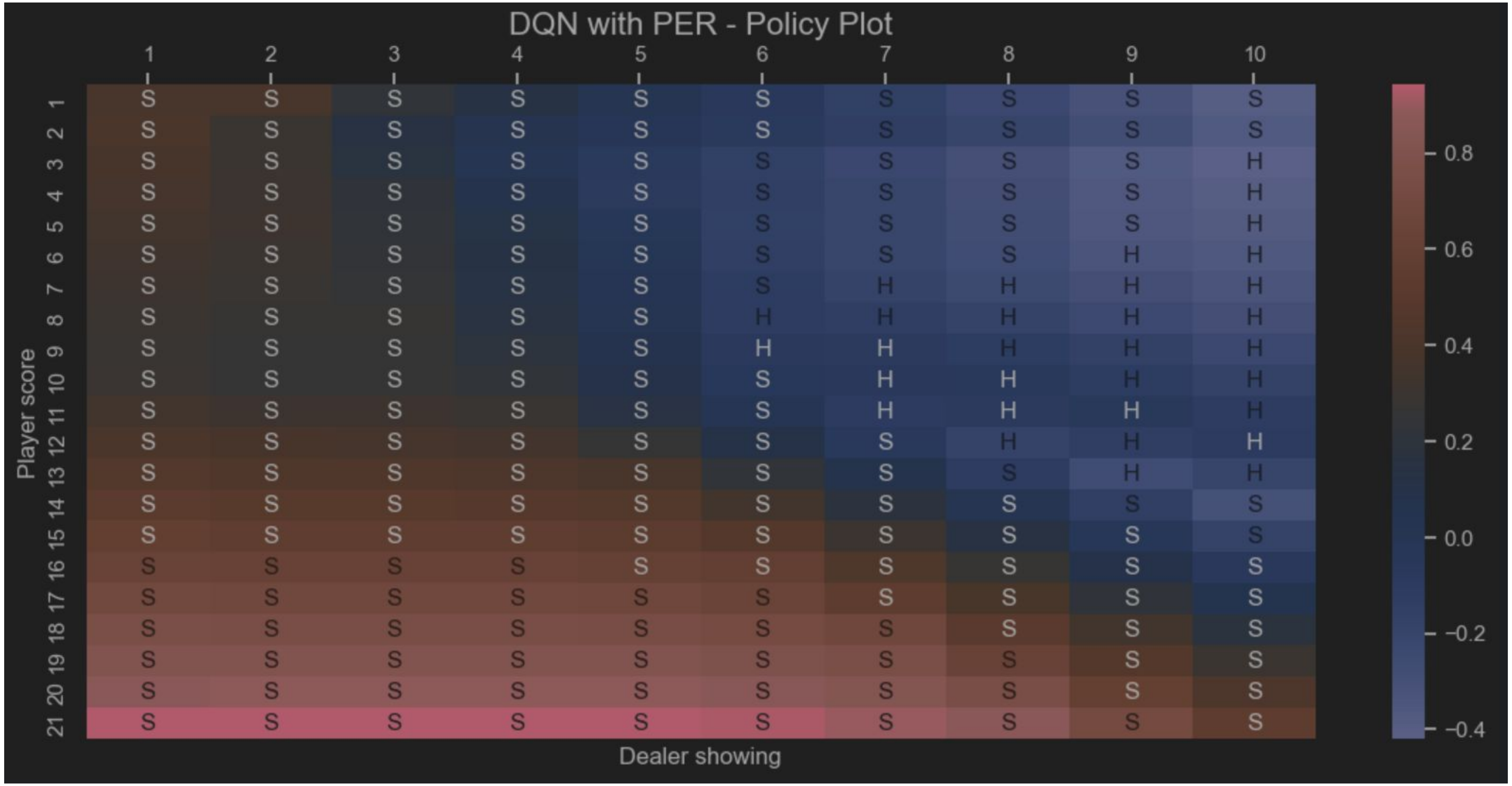


DQN + PER



Cumulative Rewards - DQN with PER
Cumulative Reward
Episode
0
−50
−100
−150
−200
−250
0
500
1000
1500
2000
2500
3000

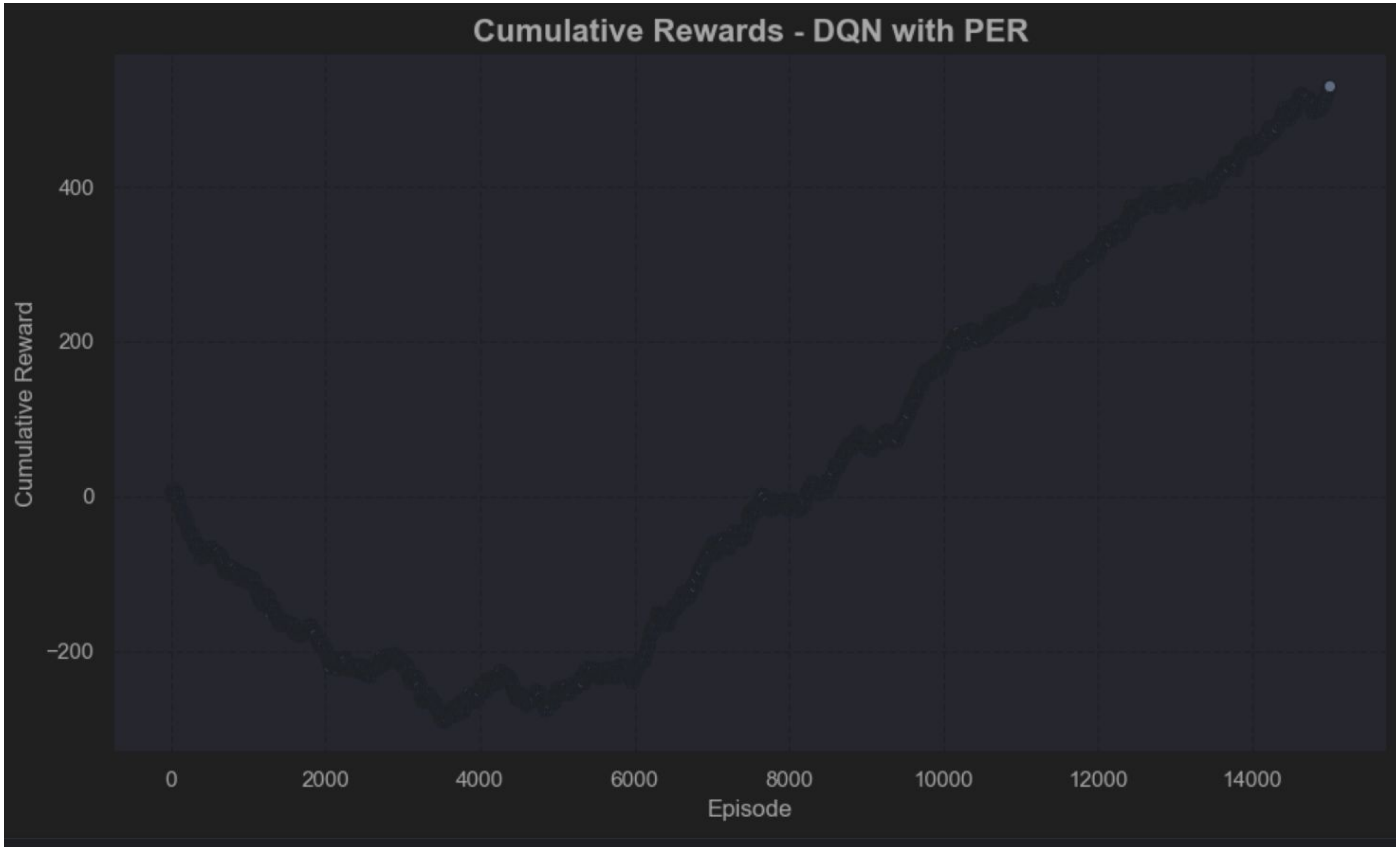
DQN + PER


DQN with PER - Policy Plot
Dealer showing
Player score
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
0.8 0.6 0.4 0.2 0.0 −0.2 −0.4

Hyperparameters

alpha_list = [0.2, 0.6]
beta_start_list = [0.4, 0.6]
lr_list = [1e-3, 5e-4]
batch_size_list = [32, 64]

DQN + DQN PER with HP



Results

1. Monte Carlo (MC)

- **How it plays**: Mostly chooses to "stick", especially when the player's score is 16 or higher.
- **Style**: Very safe, avoids risks but doesn’t take advantage of situations where the dealer has weak cards.
- **Downside**: Too cautious and doesn’t maximize potential wins.
- **Speed**: 20 ms

2. PPO (Proximal Policy Optimization)

- **How it plays**: Mixes "hit" (take another card) and "stick" depending on the situation. It’s more flexible.
- **Style**: Adapts well when the dealer has weaker cards (like 4-6), taking calculated risks.
- **Downside**: Sometimes inconsistent and takes risks when it’s not always necessary.
- **Speed** 19m 51s

3. DQN (Deep Q-Network)

- **How it plays**: Hits more often when the player’s score is around 10-14 and sticks at higher scores.
- **Style**: Balances risks and rewards, adjusting well to the dealer’s cards (e.g., hits more when the dealer has a strong hand like 7-10).
- **Downside**: Can stop too early in some situations where taking another card would have been better.
- **Speed**: 58s

4. DQN + PER (Prioritized Experience Replay)

- **How it plays**: Similar to DQN but takes smarter risks
- **Style**: A bit more aggressive, making it more effective against strong dealer hands.
- **Downside**: Sometimes takes risks that could backfire
- **Speed**: 29s