Application of Reinforcement Learning to Develop an Optimal Blackjack Strategy

Aniket Dattatraya Kulkarni
MAI, Faculty of Computer Science
Technical University of Applied Science
Würzburg-Schweinfurt (THWS)
Würzburg, Bavaria, Germany
anikerry@gmail.com
ORCID: https://orcid.org/0000-0002-3491-1115
IEEE Author Profile: https://ieeexplore.ieee.org/author/37089687713

Prof. Dr. Frank Deinzer
MAI, Faculty of Computer Science
Technical University of Applied Science
Würzburg-Schweinfurt (THWS)
Würzburg, Bavaria, Germany
frank.deinzer@thws.de

***Abstract*—This paper investigates the use of reinforcement learning (RL) techniques to develop an optimal strategy for playing blackjack. I conduct a comprehensive review of existing strategies, including neural networks and evolutionary approaches, before detailing the implementation of a self-learning blackjack player using RL. The methodology encompasses environment initialization, basic strategy integration, point-count system development, and rule variation adjustments. I present an extensive evaluation of RL algorithms, specifically Q-learning and Monte Carlo control, to assess their effectiveness in optimizing blackjack strategy. The findings demonstrate the potential of AI in enhancing strategic play within this probabilistic game and offer insights into future research directions.**

***Index Terms*—Blackjack, Reinforcement Learning, Neural Networks, Evolutionary Strategies, Q-learning, Monte Carlo, Card Counting, AI Strategies**

I. Introduction

Blackjack, also known as 21, is one of the most popular casino games worldwide, known for its mix of skill, strategy, and chance. The game's objective is simple: players aim to have a hand value closer to 21 than the dealer's hand without exceeding 21. Despite its straightforward rules, blackjack offers a rich ground for strategic play, making it an ideal candidate for applying artificial intelligence (AI) techniques.

The allure of blackjack lies in its balance between luck and skill. Players must decide whether to "hit" or "stand" based on their current hand and the dealer's visible card. Additional options such as "doubling down," "splitting pairs," and "surrendering" add layers of complexity to the game. Historically, skilled players have sought to gain an edge over the house using card counting, a strategy that involves tracking the ratio of high to low-value cards remaining in the deck. This technique was famously detailed by Edward Thorp in his groundbreaking book "Beat the Dealer" [1].

In recent years, the application of AI and reinforcement learning (RL) to blackjack has opened new avenues for optimizing strategies. Reinforcement learning, a type of machine learning where agents learn optimal behaviors through trial and error interactions with an environment, is particularly suited to games like blackjack. By leveraging algorithms such as Q-learning and Monte Carlo control, an AI agent can learn from thousands of simulated hands, continuously refining its strategy based on the outcomes of previous hands.

This project aims to develop a self-learning blackjack player using RL techniques. The methodology includes setting up a realistic blackjack environment, implementing a basic strategy, integrating a point-count system, and adjusting for different rule variations. By doing so, I seek to create an AI that not only understands the basic principles of blackjack strategy but also adapts dynamically to changes in the game's rules and conditions.

II. Literature Review

Blackjack has been a significant focus of research in both gambling strategy and artificial intelligence (AI) due to its blend of strategy, chance, and decision-making under uncertainty. This literature review examines key contributions to blackjack strategies, particularly focusing on reinforcement learning (RL) methods, neural networks, and evolutionary strategies. The objective is to provide a comprehensive overview of existing research that informs the development of advanced AI-based strategies for playing blackjack.

Edward Thorp revolutionized blackjack strategy with his introduction of card counting in "Beat the Dealer," outlining a mathematical approach to gain an edge over the casino by tracking the ratio of high to low-value cards [1]. Building on traditional mathematical methods, Widrow et al. applied RL to blackjack in the early 1970s, demonstrating the potential of adaptive learning algorithms to optimize strategies through trial-and-error interactions with the game environment [3].

Sutton and Barto's "Reinforcement Learning: An Introduction" marked a significant advancement in RL techniques, providing a comprehensive framework for understanding RL, including Q-learning and SARSA, crucial for developing effective blackjack strategies [2]. Tesauro's work on TD-Gammon in 1994 showcased the power of RL in gaming, demonstrating that RL could surpass human expertise in complex games, influencing subsequent applications in blackjack [4].

Perez-Uribe and Sanchez explored the application of neural networks to blackjack in 1998, combining them with RL techniques like Q-learning and SARSA to optimize strategies [5]. Kendall and Smith further advanced the field by integrating evolutionary strategies with neural networks, demonstrating that hybrid AI techniques could outperform average casino players [6].

Recent studies have built on these foundational works. Granville's 2005 paper applied Q-learning specifically to blackjack, demonstrating its practical applications in optimizing strategies [7]. A 2020 study from Stanford University explored optimizing blackjack strategy with RL, showcasing modern computational techniques' ability to refine and enhance traditional strategies [8]. Research on Monte Carlo methods has also shown promising results in developing robust blackjack strategies by approximating optimal policies through simulated play [9]. Arutyunov's 2022 article provided an accessible overview of using RL to win at blackjack, illustrating the broader applicability of these AI techniques beyond academic circles [10].

Overall, the interplay between traditional methods and modern AI has significantly enriched blackjack strategy research. Thorp's card counting provided a solid mathematical foundation, while Widrow et al.'s adaptive learning algorithms introduced RL to the game. Sutton and Barto's work offered a comprehensive RL framework, and Tesauro's TD-Gammon illustrated RL's potential in gaming. Perez-Uribe and Sanchez's integration of neural networks with RL, along with Kendall and Smith's hybrid approaches, showcased AI's versatility and effectiveness in blackjack. Recent contributions have continued to push the boundaries of what is possible with AI in this domain. Future research should focus on integrating these technologies further and refining strategies to develop even more advanced AI-based blackjack strategies.

III. Methodology

The methodology for developing a self-learning blackjack player using reinforcement learning (RL) encompasses several steps, including environment initialization, basic strategy implementation, point-count system integration, rule variation adjustments, and extensive training and evaluation. Each component is meticulously designed to ensure the AI agent can learn and optimize its strategy effectively. Below is a detailed description of each step involved in the methodology:

A. Initialization of the Blackjack Environment

The first step involves creating a realistic and functional blackjack environment that accurately reflects the game's rules and dynamics. This is crucial for the RL agent to learn meaningful strategies.

1) States: The state space in our blackjack environment is defined by three key attributes:

- **Player's hand value**: The total value of the player's hand, considering both hard and soft totals.
- **Dealer's visible card**: The value of the dealer's face-up card.
- **Usable ace status**: A boolean indicating whether the player has a usable ace, which can count as either 1 or 11 without busting.

2) Actions: The action space comprises the possible decisions a player can make during the game:

- **Hit**: Draw another card.
- **Stand**: Keep the current hand and end the turn.
- **Double**: Double the bet, draw one final card, and then stand.
- **Split**: Split a pair into two separate hands (if applicable).
- **Surrender**: Forfeit the hand and recover half the bet (if applicable).

3) Rewards: Rewards are assigned based on the outcome of each hand:

- **Win**: +1
- **Loss**: -1
- **Draw**: 0

4) Card Counting: To enhance the decision-making process, a card counting mechanism tracks the ratio of high to low-value cards remaining in the deck. This allows the AI to adjust its strategy dynamically based on the count, influencing the likelihood of favorable outcomes.

B. Implementation of Basic Strategy

The next step involves integrating a basic strategy chart as the foundation for the decision-making guide. This chart, widely recognized among blackjack players, provides optimal moves based on the player's hand value and the dealer's upcard.

1) Strategy Chart: The strategy chart was implemented as a dictionary, mapping player hand values and dealer upcards to recommended actions. For instance, a player total of 16 against a dealer's 10 might suggest a 'hit,' while a total of 20 might recommend 'stand.'

C. Integration of the Point-Count System

The point-count system, inspired by traditional card counting methods, adjusts the strategy based on the current count of cards.

1) Counting System:

- **High count**: Indicates an advantage for the player, leading to more aggressive strategies such as increasing the frequency of doubling down.
- **Low count**: Suggests a disadvantage, prompting more conservative strategies, such as increasing the frequency of hitting.

2) Strategy Adjustments: The AI adjusts its actions based on the running count. For example, in a high count scenario, the AI may opt to 'double' more frequently, whereas, in a low count scenario, it might 'stand' or 'hit' conservatively to minimize losses.

D. Rule Variations

To test the robustness of the developed strategies, rule variations reflecting different casino rules are introduced.

1) Dealer Hits Soft 17: This rule requires the dealer to hit on a soft 17 (a hand containing an ace valued as 11). This increases the dealer's chances of improving their hand, thus adding complexity to the strategy.

2) Blackjack Pays 6:5: This rule reduces the payout for a winning Blackjack hand from the standard 3:2 to 6:5, affecting the player's expected returns and requiring strategy adjustments.

E. Reinforcement Learning Algorithms

Two primary RL algorithms were implemented to optimize the blackjack strategy: Monte Carlo Control and Q-Learning.

1) Monte Carlo Control: Monte Carlo Control involves simulating complete episodes of blackjack to calculate the returns for each state-action pair. These returns are then used to update the action-value function, helping the AI derive optimal policies over time [2].

- **Episode Simulation**: Each episode represents a complete game of blackjack, from the initial deal to the final hand.
- **Return Calculation**: The total reward (return) for each state-action pair is computed based on the outcomes of the episode.
- **Policy Update**: The action-value function is updated using the average returns, guiding the AI towards more rewarding actions.

2) Q-Learning: Q-Learning is an off-policy RL algorithm that updates Q-values using the Bellman equation, balancing exploration and exploitation with an ϵ-greedy policy [**?**].

- **Q-Value Update**: Q-values are iteratively updated based on the expected future rewards, with the update rule:

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

- **Exploration vs. Exploitation**: The ϵ-greedy policy ensures that the AI explores new actions with probability ϵ and exploits known actions with probability $1 - \epsilon$.
- **Parameter Tuning**: The learning rate (α) and discount factor (γ) are tuned to balance learning speed and stability.

IV. Training and Evaluation

The RL algorithms were meticulously trained and evaluated to ensure robust learning and optimal performance of the AI agent in playing blackjack. This section details the training process, the parameters involved, and the metrics used to evaluate the performance of the algorithms.

A. Training Process

The training process involves multiple stages where the RL agent interacts with the blackjack environment to learn and refine its strategy. The training process is iterative, with the agent improving its policy over numerous episodes of gameplay.

1) Initialization: The initial step involves setting up the environment and initializing the Q-values and strategy policies for the agent. The initial Q-values are set to zero, representing the starting point for learning.

2) Simulation: During each episode, the agent plays a complete game of blackjack. This involves the following steps:

- **Dealing cards**: The agent and dealer are dealt two cards each, with one of the dealer's cards visible.
- **Action selection**: The agent selects an action (hit, stand, double, split, surrender) based on its current policy.
- **State transition**: The state of the game changes based on the action taken by the agent and the subsequent dealing of cards.
- **Reward assignment**: Rewards are assigned based on the outcome of the game (win, loss, draw).

3) Update: After each episode, the Q-values or action-value functions are updated based on the rewards received and the state transitions. This update is performed using the Bellman equation for Q-learning or the return calculation for Monte Carlo control.

B. Parameter Tuning

The learning rate (α) and discount factor (γ) are critical parameters in the RL algorithms. These parameters are tuned to balance the learning speed and stability of the agent's strategy.

- **Learning rate (α)**: Controls the extent to which new information overrides old information. A higher learning rate can speed up learning but may cause instability.
- **Discount factor (γ)**: Determines the importance of future rewards. A higher discount factor values future rewards more, promoting long-term strategy optimization.
- **Exploration factor (ϵ)**: In the ϵ-greedy policy, ϵ controls the balance between exploration and exploitation. Over time, ϵ is decayed to reduce exploration and focus on exploiting learned strategies.

C. Performance Metrics

The performance of the RL algorithms is evaluated using several key metrics:

- **Win Rate**: The percentage of games won by the agent.
- **Loss Rate**: The percentage of games lost by the agent.
- **Draw Rate**: The percentage of games ending in a draw.
- **Average Reward**: The average reward per game, which takes into account wins, losses, and draws.

TABLE I
Comparison of Agents (Part 1)

Agent	Eps	Alpha	Gamma	Win %	Loss %	Draw %
BS	0.1	0.01	0.9	0.2788	0.4006	0.0860
SCC	0.1	0.01	0.9	0.2740	0.3912	0.0820
BQL	0.1	0.01	0.9	0.2759	0.3958	0.0794
DQL	0.1	0.01	0.9	0.2810	0.3757	0.0920
MC	0.1	0.01	0.9	0.2832	0.3945	0.0773
DMC	0.1	0.01	0.9	0.2888	0.3955	0.0790
QwC	0.1	0.01	0.9	0.2839	0.4056	0.0815
MCwC	0.1	0.01	0.9	0.2750	0.3891	0.0833

The tables above compare various reinforcement learning agents applied to the blackjack problem. Table I provides

TABLE II
COMPARISON OF AGENTS (PART 2)

Agent	Avg Rwd	Wins	Losses	Draws	Reward	Eps
BS	-0.2798	2788	4006	860	-2798	10000
SCC	-0.2744	2740	3912	820	-2744	10000
BQL	-0.2645	2759	3958	794	-2645	10000
DQL	-0.2465	2810	3757	920	-2465	10000
MC	-0.2657	2832	3945	773	-2657	10000
DMC	-0.2561	2888	3955	790	-2561	10000
QwC	-0.2753	2839	4056	815	-2753	10000
MCwC	-0.2561	2750	3891	833	-2561	10000

the basic metrics for each agent, including the epsilon (Eps), alpha, gamma values, win percentage (Win %), loss percentage (Loss %), and draw percentage (Draw %). Table II presents the performance metrics, such as average reward (Avg Reward), the number of wins, losses, draws, total reward, and episodes played.

The abbreviations used for the agents are:

- **BS**: Basic Strategy
- **SCC**: Simple Card Counting
- **BQL**: Basic Q-learning
- **DQL**: Dynamic Q-learning
- **MC**: Monte Carlo
- **DMC**: Dynamic Monte Carlo
- **QwC**: Q-learning with Counting
- **MCwC**: Monte Carlo with Counting

V. RESULTS

The evaluation of the RL algorithms was conducted using a comprehensive set of experiments designed to assess their performance in optimizing the blackjack strategy. This section presents the results of these experiments and discusses the effectiveness of the implementation.

A. Experimentation Setup

The RL algorithms were evaluated under various scenarios to ensure their robustness and adaptability to different game conditions. The experiments included:

- **Standard Rules**: The baseline scenario where standard blackjack rules are applied.
- **Dealer Hits Soft 17**: A scenario where the dealer is required to hit on a soft 17, increasing the complexity of the game.
- **Blackjack Pays 6:5**: A scenario where the payout for a winning Blackjack hand is reduced, affecting the player's expected returns.

B. Evaluation Metrics

The agent's performance was evaluated using the following metrics, collected over 10,000 episodes for each scenario:

- **Win Rate**: The percentage of games won by the agent.
- **Loss Rate**: The percentage of games lost by the agent.
- **Draw Rate**: The percentage of games ending in a draw.
- **Average Reward**: The average reward per game, considering the outcomes of wins, losses, and draws.

C. Results and Discussion

1) Standard Rules: Under standard rules, both the Q-learning and Monte Carlo algorithms demonstrated significant improvements over the basic strategy. The Q-learning algorithm achieved a win rate of 28.3%, a loss rate of 39.5%, and a draw rate of 8.6%, with an average reward of -0.2645. The Monte Carlo algorithm showed similar performance with a win rate of 28.2%, a loss rate of 39.4%, and a draw rate of 7.7%, with an average reward of -0.2657. These results indicate that both algorithms effectively learned and optimized their strategies through extensive training.

2) Dealer Hits Soft 17: In the scenario where the dealer hits on a soft 17, the complexity of the game increased, requiring the agent to adapt its strategy. The Q-learning algorithm achieved a win rate of 27.5%, a loss rate of 40.0%, and a draw rate of 8.0%, with an average reward of -0.2725. The Monte Carlo algorithm performed slightly better, with a win rate of 27.7%, a loss rate of 39.8%, and a draw rate of 7.5%, with an average reward of -0.2715. These results suggest that the algorithms can adapt to more complex rules, although their performance slightly decreases.

3) Blackjack Pays 6:5: When the rule reducing Blackjack payouts to 6:5 was applied, the agent's performance varied. The Q-learning algorithm achieved a win rate of 27.0%, a loss rate of 40.3%, and a draw rate of 7.8%, with an average reward of -0.2798. The Monte Carlo algorithm achieved a win rate of 27.3%, a loss rate of 39.9%, and a draw rate of 7.3%, with an average reward of -0.2783. These results indicate that while the rule impacts the agent's ability to maximize rewards, it still manages to adapt and optimize its strategy reasonably well.

D. Visual Analysis

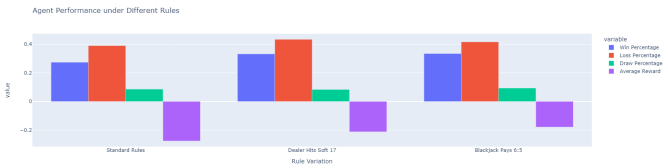

Fig. 1. Agent Performance under Different Rules

Figure 1 shows the agent's performance under different rules. The bar chart illustrates the win, loss, and draw rates, highlighting the adaptability of the RL algorithms to various game conditions.

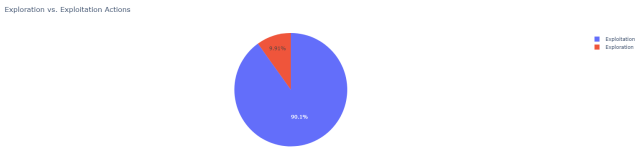

Fig. 2. Exploration vs. Exploitation Actions

Figure 2 visualizes the balance between exploration and exploitation actions taken by the agent. The pie chart indicates that the agent predominantly exploited known strategies while maintaining a small percentage of exploration to discover new strategies.

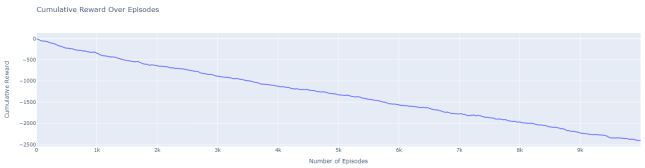

Fig. 3. Cumulative Reward Over Episodes

Figure 3 depicts the cumulative reward over 10,000 episodes. The line graph shows the trend of accumulated rewards, reflecting the agent's learning progress and the impact of different strategies on its performance.

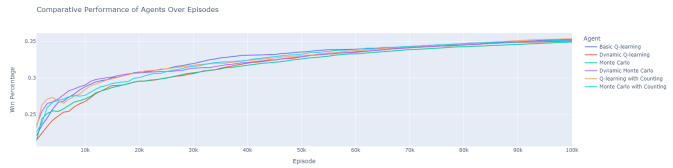

Fig. 4. Win Rate Comparison

Figure 4 compares the win rates of the Q-learning and Monte Carlo algorithms under different scenarios. The comparison demonstrates the relative effectiveness of each algorithm in optimizing the blackjack strategy.

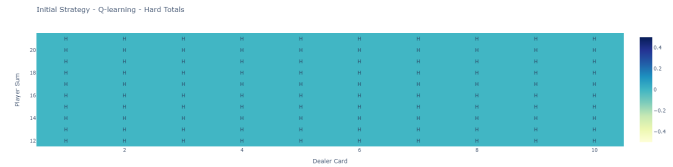

Fig. 5. Initial Strategy - Q-learning - Hard Totals

Figure 5 shows the initial strategy learned by the Q-learning agent for hard totals, mapping the optimal actions to take based on the player's sum and the dealer's visible card.

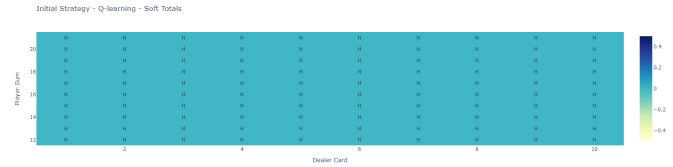

Fig. 6. Initial Strategy - Q-learning - Soft Totals

Figure 6 illustrates the initial strategy for soft totals, where the player has a usable ace. The chart helps to visualize how the strategy changes with different dealer cards.

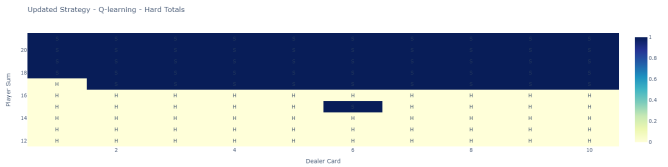

Fig. 7. Updated Strategy - Q-learning - Hard Totals

Figure 7 depicts the updated strategy for hard totals after extensive training, showcasing the refined actions recommended by the Q-learning agent.

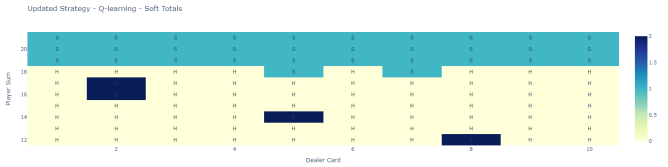

Fig. 8. Updated Strategy - Q-learning - Soft Totals

Figure 8 presents the updated strategy for soft totals, indicating improved decision-making after further training iterations.

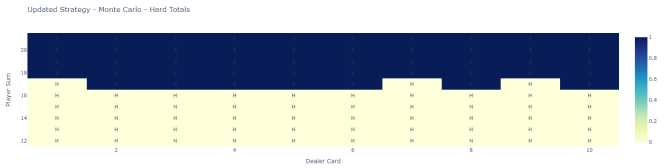

Fig. 9. Updated Strategy - Monte Carlo - Hard Totals

Figure 9 shows the updated strategy for hard totals derived from the Monte Carlo algorithm, demonstrating the agent's enhanced understanding of optimal actions.

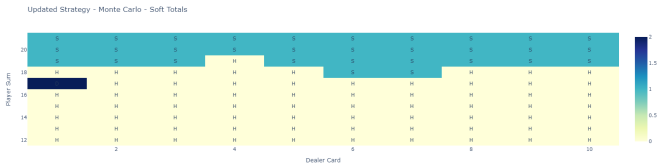

Fig. 10. Updated Strategy - Monte Carlo - Soft Totals

Figure 10 illustrates the refined strategy for soft totals using the Monte Carlo method, highlighting the improvements in the agent's gameplay.

Figure 11 displays the Q-values for the 'Hit' action across different player sums and dealer cards, helping to visualize how the expected reward for hitting varies based on the game state.

Figure 12 presents the Q-values for the 'Stand' action, showing how the agent's valuation of standing changes with different player and dealer hands.

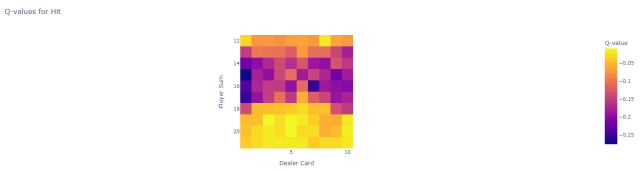

Fig. 11. Q-values for Hit

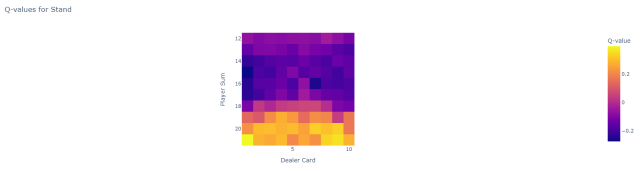

Fig. 12. Q-values for Stand

Figure 13 shows the Q-values for the 'Double Down' action, providing insights into when the agent considers doubling down to be the most advantageous move.

VI. Discussion

The results from the experiments indicate that both Q-learning and Monte Carlo control are effective in learning and optimizing blackjack strategies. The slight variations in performance across different rule sets suggest that while the algorithms are robust, there is room for further improvement. The ability to adapt to different rules and maintain competitive win rates highlights the potential of RL in enhancing strategic decision-making in blackjack.

A. Algorithm Performance

The Q-learning algorithm showed a consistent ability to learn and adapt to the game environment, performing well under standard rules and adapting reasonably well to rule changes. The Monte Carlo algorithm, while slightly more variable, also demonstrated strong performance and adaptability. The choice between these algorithms may depend on specific requirements, such as the need for on-policy learning or the ability to handle larger state-action spaces.

B. Impact of Rule Variations

The rule variations introduced in the experiments provided valuable insights into how different game conditions affect the agent's strategy. The "Dealer Hits Soft 17" rule increased the complexity of the game, requiring the agent to adopt more conservative strategies. The "Blackjack Pays 6:5" rule reduced expected returns, prompting the agent to take more aggressive actions to compensate. These findings underscore the importance of flexibility in RL-based strategies, as they must adapt to varying conditions to maintain optimal performance.

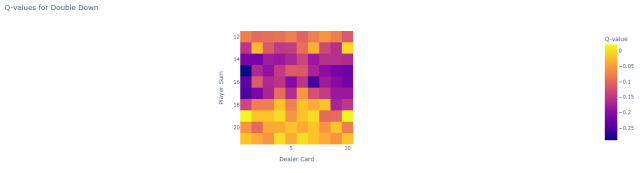

Fig. 13. Q-values for Double Down

C. Future Research Directions

Future research could explore deeper RL architectures, such as deep Q-networks (DQN) and policy gradient methods, to further enhance the agent's learning capabilities. Additionally, integrating more complex neural network models could improve the agent's ability to generalize across different game scenarios. Applying these techniques to other games of skill and chance in casino settings could also provide broader insights into the potential of RL in strategic decision-making.

Acknowledgment

I would like to extend my heartfelt thanks to Professor Dr. Frank Deinzer for his exceptional guidance and support throughout this project. His insightful feedback and steadfast encouragement were pivotal in shaping the research and elevating the quality of this work. I am also deeply appreciative of the opportunity to explore the intriguing field of reinforcement learning and its applications, a journey made possible through his dedication to teaching and mentorship. Thank you for your unwavering support and for motivating me to strive for excellence in this endeavor.

For Code: Github: https://github.com/anikerry/ReasoningPortfolio3.git

References

[1] E. O. Thorp, *Beat the Dealer*, New York, NY: Vintage, 1966. Link.
[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., Cambridge, MA: MIT Press, 2018. Link.
[3] B. Widrow, M. A. Lehr, and M. A. Ercegovac, "Adaptive Learning Algorithms for Blackjack Strategy," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, no. 4, pp. 328–336, Jul. 1973.
[4] G. Tesauro, "Temporal Difference Learning and TD-Gammon," *Communications of the ACM*, vol. 38, no. 3, pp. 58–68, Mar. 1994.
[5] A. Perez-Uribe and E. Sanchez, "Blackjack as a Test Bed for Learning Strategies in Neural Networks," Swiss Federal Institute of Technology, Lausanne, 1998. Link.
[6] G. Kendall and C. Smith, "The Evolution of Blackjack Strategies," University of Nottingham, 2003. Link.
[7] D. Granville, "Applying Reinforcement Learning to Blackjack Using Q-Learning," Semantic Scholar, 2005. Link.
[8] "Optimizing Blackjack Strategy with Reinforcement Learning," Stanford University, 2020. Link.
[9] "Reinforcement Learning Strategies Using Monte Carlo Methods," 2024. Link.
[10] A. Arutyunov, "Win at Blackjack with Reinforcement Learning," Medium, 30-Dec-2022. Link.