Comparative Study on Q-learning and SARSA For Optimal Policy Derivation for Blackjack

Bhuvana Chandra Thota
Master Artificial Intelligence, Faculty of Computer Science
Technical University of Applied Sciences Würzburg-Schweinfurt
Würzburg, Germany
bhuvanachandra.thota@study.thws.de

Abstract—**Blackjack is a card game common in worldwide casinos. An aim of the game is to acquire cards that sum up to a high number as possible that comes close to 21 but does not exceed 21. Players challenge the dealer. The game is entertaining and strategical. Blackjack is however one of the more rewarding games in a casino by a player who plays his hand carefully. In the following paper, the possibilities of Reinforcement Learning use during a Blackjack game will be described. An important aspect of winning in Blackjack is the decision that should be made during the game concerning hit, stand, double down and split and surrendering based on the situation of the game at hand. This paper will involve study of two different Reinforcement Learning algorithms namely Q learning and SARSA. Our RL agent has trained in a Blackjack (based on the standard game rules) environment with variation of a few strategies, such as the Basic Strategy, the Complete Point Count System and variations of the rules. We will train the agents on both Q-learning and SARSA and pit the two against the same environment to determine the effect the winning percentages of the agents using the complete point-counting method as compared to basic strategy. The findings are expected to assist us in comprehending the variations in practice regarding Reinforcement Learning and show us how reinforcement learning could be used to enhance strategic thinking and tactics in problematic situations.**

Index Terms—**Reinforcement Learning, Blackjack, Q-learning, SARSA,Basic Strategy,Complete Point Count System,Rule Variations.**

I. INTRODUCTION

Blackjack is also among the most researched card games in the perspective of probability and decision science. The reason behind this is that three important elements are incorporated into it, which are game of chance, player choices and well defined strategies. To illustrate, for decades the on the surf basic strategy and the card counting techniques [1] have been the de-facto standard during the decision making processes in the Blackjack game. Nevertheless, new achievements in Reinforcement Learning create new opportunities to discover data driven, dynamic solutions to play Blackjack.

In this paper we show a bunch of experiments each of which takes a popular RL algorithm, both Q-learning and SARSA, to train an agent in a simulated blackjack environment with both the Basic strategy and the varying rules in Basic strategy and a point count system and the varying rules in point count system alike. In particular, the Black jack environment based on basic strategy, and the environment of basic strategy plus configurable rules that include enabling dealer hit soft 17 and allowing disable double down are given. The second experiment, which is to practice a point-count technique and independent configuration of the expected variable rule sets, such as dealer hitting soft 17 and disable double down, coupled with the basic strategy [1].Testing and comparing performance of agents trained with RL with the different strategies is done through the experiments. They also show the dynamics of learning that can be employed to improve decision making and winning rates.

A. Blackjack

The blackjack game is one of the most famous and popular games of the casino played by people all over the world. It is a game of chances and skills in which one or more players compete against a dealer. The game goal is that a player needs to bring the value of his hand as near to 21 as possible, not to go over 21; in this case he/she will bust and can no longer win the game.Two cards are first dealt face up to each player and the dealer gets the first card face up and another face down. The face value is counted as the number of the card, face cards amount to ten and an Ace is one or eleven whichever the player likes. A player may either stick after the first deal by standing or hit, which is accepting additional cards. The dealer plays last of all the players and hits until he has a hand that equals 17 or more. The other participants are awarded when the dealer busts and the value of the hand is more than 21.In the general game players do not compete against one another, but they play against the dealer.

B. The Basic Strategy and Complete Point-Count System

The book by Thorpe [1] contains two widely known tricks on how to play Blackjack more intelligently the Basic Strategy and the Complete Point Count System. The Basic Strategy is a table of instructions which gives the precise details as how the player should play each hand so as to reduce the advantage of the casino. The strategy indicates when the player should always hit, stand, split or should always double depending on what cards they know at the time and the card that was exposed by the dealer. The approach is neither luck nor intuition based, but it is founded on the pure mathematical calculations.

The Complete Point Count System is a so called card-counting system that has an assigned value of each card. The counting rules in this system are; small cards (2 to 6) = +1;

high cards (10, and A) =-1; neutral cards (7 to 9) = 0. Being able to have a running count and true count of how many decks are left in the shoe the player can ease back to know how many tens and aces are left in the shoe. The more the tens and aces left, the higher are the opportunities of the player to win a game or even perform a Blackjack. With this information a player can adjust his bets or his plays.

C. Reinforcement Learning for Blackjack

Reinforcement Learning (RL) [2] is a sub domain of artificial intelligence where by a piece of software known as an agent train to perform optimally in a particular environment based on trial and error. The agent receives rewards or punishments depending on what it does and whether it acted well or not, and in course of time it learns the actions that can help it to achieve the best results. It has been demonstrated that it is possible to use the concept of RL in a vast number of domains in the real world, varying between robotics and games.

Blackjack is a casino game of interest and in it, one is to win the game by getting a score that is higher than the dealer sum, but not over 21. Research has also been going on widely in this field and the highly recognized methods of bettering one's chances in the game are the, Basic Strategy and Complete Point Count. As these strategies, are from the book [1], show a player can gain an advantage provided that he/she keeps a record of the cards which have been played and alter his/her action according to this fact. The open issue in this study is the question of whether a computer agent may learn how to blend these and other tactics or even become better than they and come up with its own action as it plays through millions of hands of blackjack and changes its actions automatically. An attempt to answer this question is carried out in this project by training agents to play Blackjack on Q learning and SARSA. The agents will then be subjected to an environment that is representative of the real world of Blackjack which contain all the strategies and systems in which they will be given the chance to unfold their own strategy by learning as they go along as shown in Fig 1 . The project will contain the description of the environment, overview of the learning algorithms and discussion of the results.

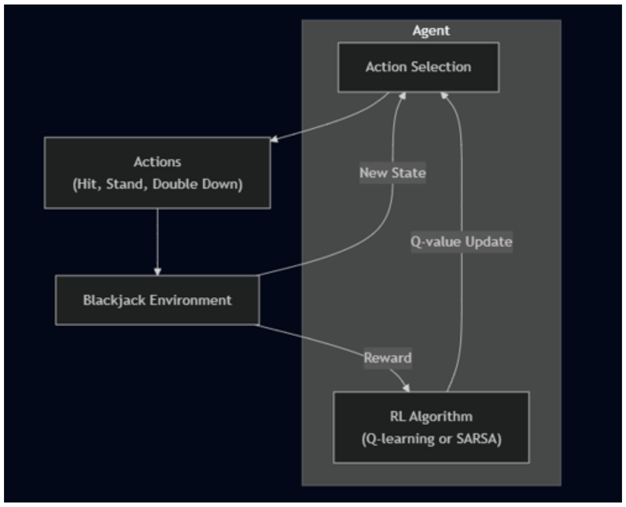

Fig. 1. Reinforcement Learning Work Flow.

D. Q-Learning and SARSA

In this research we applied the Reinforcement Learning (RL) techniques of Q-learning and SARSA [2]. Let's see the difference between them. Both of them are normally applied to tabular settings and have the same working principle of guiding an agent to predict the most valuable action by updating the table of values (Q-table) in iteration.

The fact that Q-learning is off-policy where as SARSA is on-policy is one of the specific differences between these two [2]. Off-policy implies that the agent should train the value of the best policy without taking into consideration the policy it is playing as in this instance enables the agent to estimate the best action-value function assuming that the agent would choose to make the action that has the highest Q-value [3].Contrastingly, SARSA is on-policy implying that the actions chosen by the agent involve direct impacts on its Q-value updates. This has the consequence that, in certain instances, a more conservative strategy that resulted out of SARSA would be achieved.

Each of the RL methods that have guided us to explore Blackjack enabled us to compute an expected value of a reward depending on each potential action and each potential state that is presented to an agent. Such information have been employed in the comparison of the relative strength of the various rules and strategies as it enables a learning agent to build an improvement of its play depending on the expected payback of some particular decision.

II. Methodology

In this project we have trained a reinforcement learning (RL) agent [2] to play the game Blackjack in a better way through famous Q-learning and SARSA algorithms. The method can be divided into three parts: The Blackjack Environment, the Q-learning training loop and the SARSA training loop. It creates an environment that is modified to support some of the features from the book [1]: Basic strategy, Hi-Lo point counting and modifications to the rules as possible to have a dealer hitting on soft 17, and disable double down.Each part will be explained in following subsections.

A. Blackjack Environment

The custom BlackjackEnv class implements standard rules of Blackjack that can be used in experiments with reinforcement learning. The game starts with a complete deck of cards 2-10, face cards (valued at 10) and Aces (valued at 11). Prior to every game, the deck is shuffled. Each player and dealer are dealt two cards but only one dealer card is shown to the player and is part of the state. The environment keeps a record of whether a hand is soft, i.e. whether it holds an Ace that counts as 11, and changes the Aces accordingly to avoid busting. When the value of all the hands is more than 21, then the Aces are subtracted by 10 until the hand value is below 21 or there are no Aces left. The returned state consists of total value of cards of the player, up-card of dealer, whether the hand is soft, and whether Double Down is permitted. The Double Down rule allows a player to double his bet, draw one card, and automatically stand, provided that he has two cards only. When Double Down is hit in an illegal way, it is considered as a regular Hit. The step function is used to deal with actions: Hit, Stand, or Double Down. Once the player busts and goes over 21, he or she loses automatically. When the player stands, the dealer shows his or her hidden card and has to hit until he or she has at least 17. In case the dealer busts, the player automatically wins. Otherwise, the last result is determined by the comparison of the value of hands. A victory earns one point, a defeat earns minus one and a draw earns zero. When Double Down is used, the rewards are doubled. This setting separates important Blackjack mechanics such as dynamic Ace management and Double Down and enough to allow successful reinforcement learning training with actual blackjack game.

B. Basic Strategy with Random Action

We have developed a baseline model that will give us a defined, fixed strategy player to act as a baseline against which to assess the learning-based agents. The model is utilized as a control to experiments because it provides a performance baseline that is independent of learning. Its tactic is a combination of a familiar heuristic and a little bit of random action choice, creating a powerful but unvarying adversary. The rationale of this approach is to equip the experiment with a well-defined performance measure that would allow quantifying any progress achieved by learning algorithms.

The experiment was conducted in a simulated Blackjack environment under specified rules.It used a four deck shoe which was reshuffled with every game. The dealer was instructed to stand on every 17. The game state is perceived by the agent to be a tuple composed of the hand value of the player, the upcard of the dealer, a soft ace flag, and a flag that indicates whether the player may double down. Its strategy revolves around the Basic Strategy by Edward O [1]. Thorp which is a precomputed database of the optimal decisions. This is controlled by an 0.1 -greedy strategy. This implies that the agent exploited (Thorp method) in 90 percent of the cases and explored (did some random action) in 10 percent of the cases the results of which are presented in the next section.

C. Basic Strategy with Q-learning

A Q-learning agent was applied to learn an optimal policy through direct interaction with the environment to develop an adaptive strategy. The aim of the first experiment, conducted with standard Blackjack rules and basic strategy [1], was for the agent to learn a function of action values, $Q(s, a)$, that approximates the expected return of each state-action combination. It used an ϵ-greedy strategy where the exploration parameter, ϵ, declined from 1.0 to 0.02 over 100,000 episodes. Another notable feature of this policy was the tie-breaking mechanism: the agent would default to the action suggested by Basic Strategy [1] in cases where there was more than one

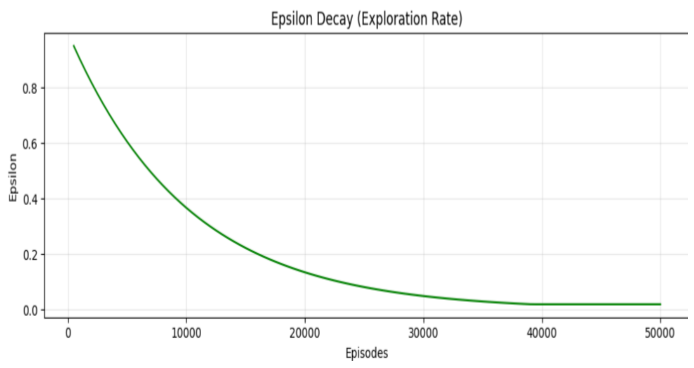

Fig. 2. The epsilon decay graph

action with the highest Q-value. A learning rate ($\alpha = 0.1$) and a discount factor ($\gamma = 0.95$) were used As you can see in Figure 2, the exploration rate starts high and gradually decreases and at each step to update the Q-table using the standard Q-learning update rule.This equation updates the Q-value of taking action a in state s $Q(s, a)$- the quality of taking action. It decreases the old value by learning rate (α) depending on the immediate reward (r) and discounted (γ) maximum expected reward of the next state and gradually discovers the optimal policy.

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)] \quad (1)$$

To test the flexibility of the agent, a second experiment was conducted in which the environment rules were modified. Specifically, two of the most popular rule changes were introduced: the Dealer Hits on Soft 17 (H17) and the prohibition of the Double Down action. Afterwards, the Q-learning agent was trained in this new environment using the same algorithm and hyperparameters as in the first experiment. This test aimed to determine whether the agent could discard the policy it had learned under the standard rules and develop a new, effective strategy adapted to the more complex dynamics.

D. Basic Strategy with SARSA

The SARSA (State-Action-Reward-State-Action) agent was developed to compare it with an off-policy model [4]. SARSA is an on-policy algorithm, meaning it learns the value of the policy it is following at any given time, including any exploratory actions. In the initial experiment in the standard environment, the SARSA agent employed the same action selection policy as the Q-learning model: an ϵ-greedy policy that similarly decays ϵ over time and uses the Thorp Basic Strategy [1] to resolve ties. The primary distinction is that it uses the actually chosen action a' in the next state s' in its update rule. The agent was initially trained using an on-policy update (100,000 episodes, learning rate $\alpha = 0.1$, and discount factor $\gamma = 0.95$):

$$Q(s, a) \leftarrow Q(s, a) + \alpha\Big[r + \gamma Q(s', a') - Q(s, a)\Big] \quad (2)$$

To test the flexibility of this on-policy algorithm, the SARSA agent was also trained with rule variations, creating a more challenging environment with the Dealer Hits on Soft 17 (H17) and No Double Down rules. All hyperparameters remained the same as in the first experiment, as did the algorithm, leaving the changes in the rules as the only variable in order to clearly evaluate the algorithm's ability to develop a new policy. The primary goal was to compare the adaptive capability of SARSA with the Q-learning agent and provide insight into the practical differences between on-policy and off-policy learning in this specific problem domain.

E. Point Count System Environment

So, to find out whether we could train a computer to play Blackjack as an expert, we had to create our own digital version of the game. That provided us with a environment where we could experiment with our ideas perfectly well. We also established our virtual casino so as to play with a four-deck shoe, which was re-shuffled whenever a new hand occurred. The dealer was also programmed to follow a house rule that was common: he/she had to stand on any hand that was valued 17 whether it was a hard total or soft total with an Ace. Our learning agent had the same capabilities as a person in playing the game, where it could hit, stand or double down. Naturally, the doubling down was only possible on its initial two cards. The agent had to know how to determine what worked by getting clear feedback on its performance. That is why we created an easy yet efficient reward system: a win brought one point, a loss brought one point, and a tie brought zero. In the event that it won a double-down bet, it was rewarded two points of course. Such simple feedback was a must so that the agent could learn the relationship between its actions and the eventual outcome of the game. The sharing of information is the key concept of our approach. We provided the agent with additional information to enable it to count cards. We did not merely teach the basic strategy, we taught the advanced strategies which the pros employ. In order to do so we introduced more information regarding the state of the game. Introduced Hi-Lo counting system [1] where we informed the agent of the actual number of cards in the deck, its own cards and the cards that the dealer displayed. This information was constructed into the environment. Whenever a card was exposed, the agent would adjust a count +1 for low cards and -1 for high cards. Then it computed the greater true count based on the number of decks remaining. Due to this complete point count strategy [1] we shared, the agent was able to concentrate on the upper level strategy: when and how to bet. It is the same decision that the pro players make, thus it appears that the agent possessed a professional feel of the game.

F. Point Count System with Q-Learning

We investigate the extent to which an agent can learn a point-counting strategy [1] via Q-learning. The algorithm is applied to two Blackjack environments and the results are compared. [3]Q-learning is an off-policy, model-free algorithm that approximates the value of each action in any given state. The estimate is constructed through the development of a Q-table. Whenever a potential move is made, the algorithm updates the relevant entry using the standard Q-learning equation in equation 1 above.The agent in both environments has an augmented state representation. This representation provides the agent with three types of information at any given time: the True Count of the Hi-Lo system, the player's hand, and the dealer's upcard. With this information, the agent can learn a strategy that not only plays the game but also adapts to changes in the deck composition.

In the first experiment, the agent plays a standard Blackjack game in which the dealer must stand on 17 and the player may double on the first hand. This setup favors the player and serves as a useful initial case for studying how the agent develops and improves a simple point-counting strategy.

The second experiment tests the agent's ability in a more challenging environment. In this case, the dealer stands on soft 17 (H17), which increases the house edge, and the player is not permitted to double down—a major opportunity for exploiting favorable odds. Comparing the agent's performance in the two environments helps gauge the effectiveness of Q-learning in enabling the agent to adapt its strategy when the rules and payoffs of the game are altered.

G. Point Count System with SARSA

In these experiments, we employed the SARSA algorithm. SARSA (State-Action-Reward-State-Action) is a temporal-difference, on-policy learning method [4]. It improves the agent's policy by learning directly from the actions that the agent has actually undertaken. The goal is to construct a Q-table, which estimates the value of a state-action pair, $Q(s, a)$, and this table is updated using the rule in equation 2 above.

More importantly, a' is the action that the agent will take in the following state, s', which characterizes the on-policy nature of the strategy. The success of this method is directly connected to additional information: the True Count of the Hi-Lo system. This information enables the agent to develop a card-counting plan that responds to changes in the deck composition. This setup was first tested under ordinary Blackjack rules: the dealer stands on all 17s, and the player is allowed to double down. This case served as a benchmark to examine how well SARSA could learn a point-counting strategy in a standard setting.

A second experiment was then conducted in a more challenging situation. The rules were modified so that the dealer stands on soft 17 (H17) and the player is not allowed to double down. This arrangement is less favorable for the player and allows us to observe how the SARSA agent adjusts its strategy as the game dynamics change. By comparing its performance under each set of rules, we can gauge how effectively the agent's policy adapts when the game conditions vary.

III. Evaluations and experiments

The reinforcement learning (RL) was also applied to determine whether it was functioning properly and was returning good results. A series of experiments was used to carry this

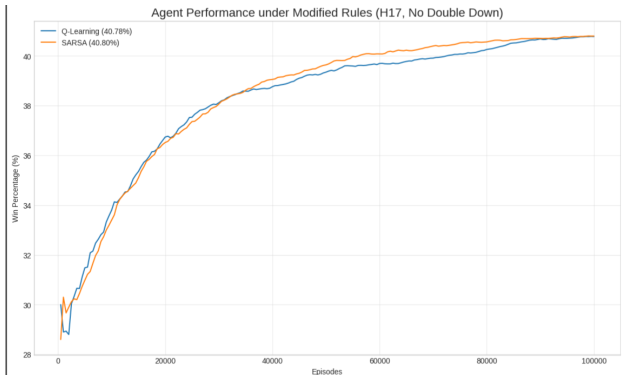

Fig. 3. Agent performance under rule variations(H17, No Double Down). Both Q-Learning and SARSA agents ver 100,000 episodes in the case of basic strategy.

out. It was aimed at observing the effect of various strategies and rules on the performance demonstrating a definite hierarchy of strategic advantage. Each run was done with three episodes 50,000, 100,000, and 1,000,000 steps.

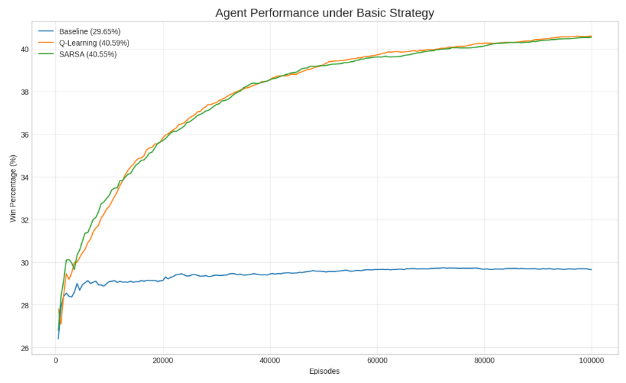

Fig. 4. Agent performance comparison between a Baseline strategy, Q-Learning, and SARSA over 100,000 episodes in the case of basic strategy.

A. Performance with Basic Strategy

We began by using a non-learning agent which employed Thorp Basic Strategy but with a 10 percent probability of executing a random action. This agent had a winning percentage of 29.71 percent of its games as you see the experiment results in table I, proving that it is not possible to obtain good performance with a simple, fixed rule set, particularly when combined with random, sub-optimal moves.

TABLE I
BASELINE PERFORMANCE (BASIC STRATEGY + 10% RANDOM EXPLORATION)

Episodes	Win Rate (%)	Loss Rate (%)	Draw Rate (%)
20,000	29.32	64.93	5.75
50,000	29.79	64.66	5.56
1,00,000	29.71	64.74	5.55

As soon as reinforcement learning was applied, a significant improvement was observed, introducing a Q-Learning and a SARSA agent. Having analysed millions of games, the agents developed a policy that significantly outperformed the baseline. Both Q-Learning and SARSA arrived at an extremely similar, better strategy and came to a win rate of approximately 40 percent As shown in Figure 4. This shows that the learning algorithms were well-taught and would be able to explore the state space of the game by themselves in order to find a very efficient policy with no assistance.

These RL agents were also trained with the more difficult variation of rules set (dealer hits on soft 17, no double down). The performance of these agents was very consistent in this situation as they won around 40.8 percent as shown in in Figure 3.This is a slight modification that is to be expected; with no finer details that basic strategy would offer, the strategy that has been learned by the agents is solid but does not contain a method of changing its strategy dramatically in reaction to these particular rule changes.

B. Performance with Complete Point Count Strategy

The second stage of the assessment employed sophisticated agents where the Hi-Lo card-counting was employed. The agents only needed to remember the true-count in their state and it was simpler to acquire a strategy that reacted to the changing balance of cards.

• Under Standard Rules: The counting agents had a prominent and pronounced advantage. The average win rate of the Q-Learning and SARSA players was around 43.7 percent. The best was for SARSA as the win rate is noted in Figure 5. This is almost two percentage points superior to the non-counting agents and establishes that the agents learned to associate the card count to good bets and exploit their advantage whenever they had a favorable deck.

• With Rule Variation : To determine how effective the

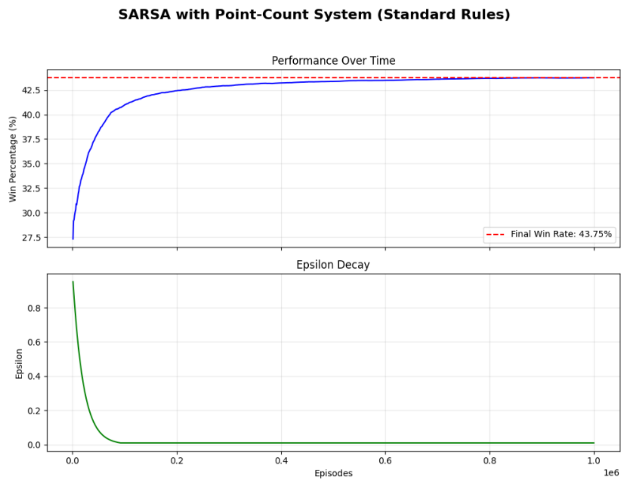

Fig. 5. Best Performance of experiment SARSA agent using a point-count system under standard rules.the bottom plot displays the epsilon decay over one million episodes.

strategy of the agent adapted to, the counting agents were made to play in a more difficult environment (dealer hits soft 17, no double down). With such a handicap they won 43.2 percent as shown in tableV. Although this is not much of a decrease in comparison to the standard-rule outcome, this score is much higher compared to the non-counting agents. This demonstrates the fact that their learned policy is capable of adapting to bigger house edges and be at play.

C. Overall Results

The following tables provide a detailed breakdown of the results at various stages of the simulations, from 20,000 to 1,000,000 episodes. The performance is clearly seen through evaluation. The agents that recorded the best win rate were point-counting agents [4]. The learning agents outdid the baseline by a big margin which proved that they were able to utilize complex information to achieve a quantifiable benefit even in challenging situations.

TABLE II
PERFORMANCE WITH BASIC STRATEGY(STANDARD RULES)

Agent	Episodes	Win Rate (%)	Loss Rate (%)	Draw Rate (%)
Q-Learning	50,000	38.96	52.92	8.12
	100,000	40.30	51.38	8.31
	1,000,000	41.81	49.18	9.01
SARSA	50,000	39.16	53.26	7.58
	100,000	40.43	51.12	8.45
	1,000,000	41.84	49.17	8.98

TABLE III
PERFORMANCE WITH BASIC STRATEGY(RULE VARIATIONS)

Agent	Episodes	Win Rate (%)	Loss Rate (%)	Draw Rate (%)
Q-Learning	50,000	39.18	53.13	7.68
	100,000	40.86	50.91	8.23
	1,000,000	41.94	49.20	8.86
SARSA	50,000	39.50	52.89	7.60
	100,000	40.98	50.95	8.07
	1,000,000	41.85	49.33	8.82

TABLE IV
PERFORMANCE WITH POINT COUNT STRATEGY(STANDARD RULES)

Agent	Episodes	Win Rate (%)	Loss Rate (%)	Draw Rate (%)
Q-Learning	50,000	37.68	55.49	6.83
	100,000	40.52	51.81	7.67
	1,000,000	**43.56**	**47.28**	9.16
SARSA	50,000	38.00	55.12	6.87
	100,000	40.64	51.55	7.81
	1,000,000	**43.75**	**47.19**	9.05

TABLE V
PERFORMANCE WITH POINT COUNT STRATEGY(RULE VARIATIONS)

Agent	Episodes	Win Rate (%)	Loss Rate (%)	Draw Rate (%)
Q-Learning	50,000	38.30	55.16	6.54
	100,000	40.67	51.83	7.50
	1,000,000	43.22	48.02	8.76
SARSA	50,000	38.31	54.96	6.73
	100,000	40.86	51.44	7.70
	1,000,000	43.20	48.08	8.72

IV. CONCLUSION

This study has been able to prove the effectiveness of the reinforcement learning in coming up with advanced winning solutions to the game of Blackjack. These findings validate that both Q-Learning and SARSA agents could successfully learn good policies, raising the win to about 42 percent as compared to the baseline of about 29.7 percent. Moreover, given the correct count of cards, the agents were able to learn how to use this knowledge to their advantage and increase their win rate to more than 43.7 percent with the point count strategy. Their adaptability was also outstanding as they were able to continue winning more often despite the rule variations that put them at a disadvantage by raising the house edge.

Although these findings are encouraging, they also create several avenues that can be explored in the future.It is also possible that additional research will investigate the use of more intricate card counting systems (e.g., Wong Halves, Omega II) and whether a more detailed count can provide a larger edge. Also, the simulation could be extended to the multi-player environment, which would challenge the strategy of the agent in a more dynamic environment, where the choices of other players affect the deck makeup. And lastly, in even more complicated state representations, moving on to Deep Reinforcement Learning might enable an agent to learn off of continuous, non-discretized data, possibly revealing more subtle and powerful strategies.

REFERENCES

[1] E. O. Thorp, Beat the Dealer: A Winning Strategy for the Game of Twenty-One. New York, NY, USA: Vintage, 1966.

[2] Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. MIT Press,Cambridge, MA, 2 edition, 2018.

[3] De Granville, C. Applying reinforcement learning to blackjack using q-learning. University of Oklahoma.

[4] Geiser J, Hasseler T. Beating Blackjack-A Reinforcement Learning Approach.