Creating an optimal Blackjack agent that explains its decision-making process for human insight

Julian Glover*, Sidhanth Kapila†, Kaamran Rizvi‡

December 2024

1 Abstract

The focus of this project is to build a Blackjack agent that is trained using the Stochastic Variance Reduction for Deep Q-learning (SVG-DQN) and a multi-Layer perceptron as its Q-function approximator. We then test our trained model on new random states spaces to get a detailed look at how it makes its decision, to hit or stand, by using a Shapley-Shubik explainability algorithm. The goal is to draw out explanations and build an understanding of how an agent learns to play Blackjack and to theorize potential broader learning opportunities and applications in game based environments.

2 Introduction

In the game of Blackjack, the player's strategy is influenced not only by its own cards (other players as well if applicable) but also the dealer's card – who they play in opposition to. The player's objective is to get as close as possible to 21 without exceeding it. Initially, a player is given two cards, the dealer is given one, and must choose to either "hit" or "stand". A hit means to take another card, changing a player's hand, whereas a stand means to decline taking another card, keeping their hand the same. After a player chooses to stand, the dealer then continues to draw cards until they reach a score of 17 or higher. At the end of a game, if your score is higher than the dealer's, you win; if it is lower, you lose; and if it matches the dealer's score, the game ends in a tie. If the player or dealer's score exceeds 21, they lose. Finding an optimal strategy to maximize a player's win requires an understanding of many factors. The game's dynamic environment and inherent probabilistic nature require a player to consider the weight of their own hand, the dealer's hand, and the cards that are left to be drawn in order to be successful. For the average player, this understanding for optimizing their win rate is difficult to obtain without continuous play or extensive research into the probabilistic breakdown of the game. The stochastic elements of the game (card draws, dealer actions) make Blackjack a compelling game environment to explore the concept of decision-making under uncertainty using artificial intelligence.

The focus for this research will be **how can we create an agent capable of playing Blackjack optimally while also understanding how it employs its strategy, in order to teach users how to improve their own decision-making?** Our proposed solution is to create an agent that can learn an optimal strategy to play Blackjack by utilizing Stochastic Variance Reduction for Deep Q-learning (SVG-DQN) and a multi-layer perceptron as it's Q-function approximator.

The second focus was to make the model's decision-making explainable. To implement this, we used the Shapley–Shubik method—an approach that assigns importance to each feature based on their contribution across all possible sequences—to account for the dynamic, sequential feature importance and interactions in our Blackjack decision-making model.

*jtglover@usc.edu
†skapila@usc.edu
‡karizvi@usc.edu

Lastly, we evaluated how well our agent performed against an algorithm that mimicked the existing optimal strategy for Blackjack and looked to the model's SHAP values to analyze its decision making process.[1]

3 Related Work

3.1 Deep Q-learning Applied to Blackjack

A student named Allen Wu from Stanford's Management Science and Engineering program approached training a Deep Q-learning algorithm, as Deep-Q learning algorithms performed well for Atari and Go. Compared to other models that were tested at the time (Traditional Q-learning), it performed much better, and the derived policy from the agent learned the game well. However, it did not learn the optimal policy of Blackjack, which was surprising to the author as the state space of Blackjack is smaller than games like Atari; they believe it has to do with the fact that deep Q-learning struggles with probabilistic transitions and rewards. Nonetheless, the author notes the model is primitive, and some work with hyperparameters along with more recent innovations (or possibly another reinforcement learning algorithm) could bring better results on Blackjack [8].

3.2 SHAP for Explainability

On the topic of explainability, a common issue faced in the field of machine learning is the problem of understanding model behavior. It's important to understand how exactly a model acts and why, which is the root of explainability. In a paper by Thomas Hickling, Abdelhafid Zenati, Nabil Aouf, and Phillippa Spencer one of the techniques they highlight to help approach this issue is Shapley Additive explanations (SHAP). The SHAP technique is rooted in game theory and is executed similarly to many other game theory problems, where features are each given values based on how important it is to the model's prediction. This is great in a game like Blackjack since there are plenty of instances where one card or action (by the dealer or other players) can have a significant effect on a player's odds, and its basis in game theory is great considering the player has (generally) a basic two-choice decision to make, hit or stand, playing as opponent to the dealer (they can't both win) [2].

3.3 SVG for Deep Q-learning

The third paper we explored is Stochastic Variance Reduction for Deep Q-Learning [9]. In Wu's paper he implemented just a Deep Q-learning algorithm, which is useful, but a newer Deep Q-Learning algorithm called Stochastic Variance Reduction for Deep Q-learning enhances traditional DQN. More specifically, it tackles the high variance in gradient estimation present in DQN, which leads to worse training and poor precision, specifically in environments like Blackjack, by introducing probabilistic transitions and threshold-based rewards. Essentially, SVG-DQN enables higher accuracy and stability for its updates, which also leads to faster convergence - this allows for the model to better handle critical thresholds like in Blackjack (good reason as to why it's better over DQN, fast convergence times). The researchers in the paper also performed their tests against the Atari arcade learning environment where it performed better than previous DQN model for game environment, with less training iterations.

3.4 MLP for Game Environments

A related paper that focused on playing Atari games [4] mentioned the use of multi-layer perceptrons to make decisions in a game setting, considering that they can adjust weights, optimize for loss, and can approximate nonlinear relationships such as those in Blackjack. They mention the use of stochastic gradient descent to minimize loss and adjust weights, though they note it's computationally expensive for more intricate games with a high number of parameters. We also used a two-layer perceptron with a hidden layer as it can model more complex, nonlin-

[1] One clarification to be made here is that "SHAP" values in our paper refer to values dervied from the Shapley-Shubik algorithm, as opposed to the SHAP technique.

ear relationships than a single-layer perceptron. The hidden layer introduces the ability to learn intermediate features, and in a game like Blackjack this is more than enough.

4 Methodology

4.1 Training the Model

We trained an agent using Stochastic Variance Reduction for Deep Q-Learning (SVG-DQN) in an attempt to get the model to converge faster to provide more stable gradient updates as a result of the stochastic variance reduction (a key component of Blackjack is drawing cards, which is stochastic, SVG-DQN helps model better deal with noise from that). Moreover, we implemented a multi-layer perceptron as our Q-function approximator over a table because it is overall more dynamic (generalizes to unseen states more efficiently). We chose to use OpenAI's gymnasium Blackjack state space, where our agent would learn to hit and stand optimally based on three data points: 1. The player's hand, 2. usable aces (if a current ace can be counted as 11 and not be over 21), and 3. The dealer's visible card. Moreover, the dealer has to hit until their hand value is greater than or equal to 17, which is common in most casinos.

The next step was to define classes for our SVG-DQN and MLP, so that we could use the SVG to train our MLP in approximating values when deciding to hit or stand. Specifically, the MLP is a sequential neural network that has two linear layers and uses ReLU activation functions. Additionally, during training, we used an epsilon-greedy policy algorithm that explored and exploited the model's learned strategy, decaying over time from a starting point of all exploration to all exploitation. For the training loop, we run 10,000 episodes, each beginning with a reset of the environment. During each episode, experiences are stored in a replay buffer as the agent interacts with the environment. Either a random action is chosen with a probability determined by the epsilon, or the action with the highest Q-value is selected. During the episode, if the replay buffer has more than 64 experiences, a random mini-batch of 64 examples is chosen and target Q-values are calculated using our target network (a copy of our network that is copied over every 100 episodes to aid in training). For Q-network updates, our Q-network predicts values for the current state and actions in mini-batch to calculate loss between these predicted values and the corresponding true values from the target network. We then use loss, gradient descent, and Adam to better update the parameters of our Q-network. Then we update the reward and transition to the next state.

4.2 Evaluation of Model

Following training, we then wanted to evaluate this trained model. To do so, we created an evaluation loop that ran for 50 trials, with each trial consisting of 1000 episodes. To evaluate the trained model, we set the epsilon value to 0, ensuring the agent relied entirely on its learned strategy without any exploration. We then tracked the win rate of our model over these periods: the average win rate being 42 percent (see Figure 1 below). For reference, the optimal win rate for Blackjack is around 42 percent [7], so our agent in fact plays near optimal for the state space we defined.

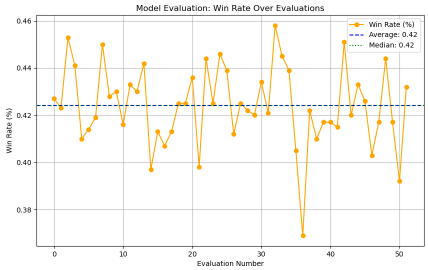

Figure 1: The win rate of the trained Blackjack agent across 50 trials shows an average win rate of 42 percent, near the optimal performance for Blackjack.

4.3 Shapley-Shubik

Once the agent was playing at an optimal level, we focused on the explainability aspect of our model. For

our algorithm, we implemented the Shapley-Shubik algorithm that takes a single state and our action (hit or stand), and computes the contribution of each feature in the state by seeing how the Q-values change as features are added and subtracted. For each permutation of the features, the marginal contribution of a feature is calculated by measuring the change in the Q-value when the feature is added compared to when it is not included. To ensure it is correctly done, contributions of each feature across all the permutations are averaged - which then gives final Shapley-Shubik values for each feature. Once completed, we then had a CSV for thousands of states and scenarios where for our 3 features - users hand, dealer's card, and usable ace - we could see how much each feature contributed to the agents' decision to either hit or stand, and if our agent won or lost.

5 Experiment Results

As mentioned above, the agent played Blackjack at a near optimal level of 42 percent. However, we also know that the state space is simpler than the space seen in the development of typical Blackjack strategy: there is no betting, actions like splitting, or other players. As a result, we wanted to establish how our agent played in comparison to the optimal Blackjack play style in our state space to get a better baseline of what an optimal win rate in our state space looks like.

5.1 Agent Model vs Standard Model

Before running the standard model, we wanted to create charts for the standard strategy and for the actions our agent took in each state (hard and soft hands) in order to compare the two for every scenario. This helps determine whether it is worth testing the baseline model, as it is quite possible that the agent learned to play the traditional strategy given its great performance, making a comparison pointless.

HARD TOTALS

	DEALER UPCARD									
	2	3	4	5	6	7	8	9	10	A
17	S	S	S	S	S	S	S	S	S	S
16	S	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
12	H	H	S	S	S	H	H	H	H	H
11	D	D	D	D	D	D	D	D	D	D
10	D	D	D	D	D	D	D	D	H	H
9	H	D	D	D	D	H	H	H	H	H
8	H	H	H	H	H	H	H	H	H	H

SOFT TOTALS

	DEALER UPCARD									
	2	3	4	5	6	7	8	9	10	A
A,9	S	S	S	S	S	S	S	S	S	S
A,8	S	S	S	S	Ds	S	S	S	S	S
A,7	Ds	Ds	Ds	Ds	Ds	S	S	H	H	H
A,6	H	D	D	D	D	H	H	H	H	H
A,5	H	H	D	D	D	H	H	H	H	H
A,4	H	H	D	D	D	H	H	H	H	H
A,3	H	H	H	D	D	H	H	H	H	H
A,2	H	H	H	D	D	H	H	H	H	H

KEY		
	H	Hit
	S	Stand
	D	Double if allowed, otherwise hit
	Ds	Double if allowed, otherwise stand
	N	Don't split the pair
	Y	Split the Pair
	Y/N	Split only if `DAS` is offered
	SUR	Surrender

Figure 2: Standard strategy through every state.

However, when comparing Figure 2 [1] to Figure 3 and 4, there seem to be differences in hitting or standing in certain states. So, to calculate the win rate for the standard strategy, we created an algorithm to test it on the same 50,000 initial hands that our agent was evaluated on. The standard model achieved a win rate of about 44 percent, slightly better than the agent. Although the agent diverges from the standard strategy at certain points, its winning rate was very close to optimal. As a result, we questioned whether the agent performed better or worse in comparison to the standard approach for the states in which the agent diverges. Also, what does the agent look at in these states to make its decision?

To address the first question, even though the standard model performs better, this does not mean the

Hit (H) vs Stand (S) by Player Hand vs Dealer Card (No Usable Ace)

	1	2	3	4	5	6	7	8	9	10
4	H	H	H	H	H	H	H	H	H	H
5	H	H	H	H	H	H	H	H	H	H
6	H	H	H	H	H	H	H	H	H	H
7	H	H	H	H	H	H	H	H	H	H
8	H	H	H	H	H	H	H	H	H	H
9	H	H	H	H	H	H	H	H	H	H
10	H	H	H	H	H	H	H	H	H	H
11	H	H	H	H	H	H	H	H	H	H
12	H	H	H	H	H	H	H	H	H	H
13	H	H	H	H	H	H	H	H	H	H
14	H	H	H	H	S	H	H	H	H	H
15	S	S	S	S	S	S	S	H	H	H
16	S	S	S	S	S	S	S	S	H	S
17	S	S	S	S	S	S	S	S	S	S
18	S	S	S	S	S	S	S	S	S	S
19	S	S	S	S	S	S	S	S	S	S
20	S	S	S	S	S	S	S	S	S	S
21	S	S	S	S	S	S	S	S	S	S

Figure 3: Model's decision through every state, usable ace is not present.

Hit (H) vs Stand (S) by Player Hand vs Dealer Card (Usable Ace)

	1	2	3	4	5	6	7	8	9	10
12	H	H	H	H	H	H	H	H	H	H
13	H	H	H	H	H	H	H	H	H	H
14	H	H	H	H	H	H	H	H	H	H
15	H	H	H	H	H	H	H	H	H	H
16	H	H	H	H	H	H	H	H	H	H
17	S	S	S	S	S	S	S	S	H	H
18	S	S	S	S	S	S	S	S	S	H
19	S	S	S	S	S	S	S	S	S	S
20	S	S	S	S	S	S	S	S	S	S
21	S	S	S	S	S	S	S	S	S	S

Figure 4: Model's decision through every state, usable ace is present.

divergences in strategies for the agent are necessarily bad, as what matters most is how it compares to the traditional strategies win rate for that given state. It is important to note, while it was the same counts for each state, how each state played out was slightly different in some cases. For example, even if both the agent and standard model hit for a certain state, the card they received was different in some scenarios - this is always true in Blackjack, a player receives different cards each time they hit, regardless of their hand. Thus, the reason for comparing the states in which they differ is that in the states in which they employ the same strategy, any differences in outcome can be attributed to the random card that is drawn. Nonetheless, we then compared the win rates between the standard model and our agent in the states in which they differed (Figures 5 and 6).

Comparison of Agent and Standard Strategy (No Usable Ace)

State	Agent Action	Standard Action	Agent Win Percentage	Standard Win Percentage	Avg Player's Hand (%)	Avg Dealer's Card (%)	Avg Usable Ace (%)	Counts
Player=12, Dealer=4, Ace=0	H	S	35.49%	23.32%	56.34849323427365	41.10379590836351	2.547710857362858	448
Player=12, Dealer=5, Ace=0	H	S	37.03%	28.32%	55.33944756220177	41.90306336289635	2.757489074901887	451
Player=12, Dealer=6, Ace=0	H	S	36.65%	33.63%	56.39609566556912	40.62230381800237	2.981600516428508	442
Player=13, Dealer=2, Ace=0	H	S	35.06%	21.94%	61.77582097771006	36.17956404221951	2.044614980070426	445
Player=13, Dealer=3, Ace=0	H	S	36.76%	24.35%	59.631076242402315	38.184754430570486	2.18416932702719	438
Player=13, Dealer=4, Ace=0	H	S	33.54%	27.30%	60.15482739071131	37.46889953364195	2.376273075646733	483
Player=13, Dealer=5, Ace=0	H	S	35.47%	30.97%	60.650388430447634	36.79184830824448	2.5577632613078913	437
Player=13, Dealer=6, Ace=0	H	S	34.97%	27.91%	60.694365190799736	36.55617775087764	2.749457058322644	449
Player=14, Dealer=2, Ace=0	H	S	32.74%	25.44%	65.4150903444805	32.65060968086896	1.934299974650515	446
Player=14, Dealer=3, Ace=0	H	S	31.37%	20.20%	63.01559686072403	34.9256598564257	2.0587432828502576	475
Player=14, Dealer=4, Ace=0	H	S	30.69%	21.29%	63.479372786999065	34.29206752037091	2.2285596926300264	479
Player=14, Dealer=6, Ace=0	H	S	34.71%	33.56%	64.40071118385788	33.04567198299822	2.5536168331438995	461
Player=15, Dealer=1, Ace=0	S	H	10.61%	19.86%	46.60401931964568	7.721940078709523	45.6740406016448	490
Player=15, Dealer=7, Ace=0	S	H	28.92%	37.22%	21.460327232988043	21.093592328649045	57.44608043836291	453
Player=16, Dealer=1, Ace=0	S	H	12.83%	22.54%	37.372359382871245	7.717069674157144	54.91057094297162	460
Player=16, Dealer=7, Ace=0	S	H	24.07%	32.90%	45.61882612154167	5.648052562938109	48.733121315520215	457
Player=16, Dealer=8, Ace=0	S	H	25.05%	31.05%	31.51743828633189	27.06772025060532	41.41484146306279	483
Player=16, Dealer=10, Ace=0	S	H	21.05%	24.80%	18.95145188984289	48.65586778268836	32.39268032746875	2043

Figure 5: Model's decision through every state, usable ace is not present.

Comparison of Agent and Standard Strategy (Usable Ace)

State	Agent Action	Standard Action	Agent Win Percentage	Standard Win Percentage	Avg Player's Hand (%)	Avg Dealer's Card (%)	Avg Usable Ace (%)	Counts
Player=17, Dealer=1, Ace=1	S	H	10.59%	47.06%	1.4900909194598717	57.78651014318953	40.72339893735059	85
Player=17, Dealer=2, Ace=1	S	H	35.38%	23.81%	48.728756474377086	19.81281730702785	31.458426218595054	65
Player=17, Dealer=3, Ace=1	S	H	40.30%	47.73%	59.88838250208068	19.55663698998322	20.55498050793609	67
Player=17, Dealer=4, Ace=1	S	H	38.89%	32.43%	67.05114376191457	8.352237862502147	24.596618375583276	54
Player=17, Dealer=5, Ace=1	S	H	37.50%	42.55%	70.5348413771131	2.024484058546782	27.440674564340117	72
Player=17, Dealer=6, Ace=1	S	H	29.58%	42.50%	69.36612446668977	6.617252194674474	24.01662333863576	71
Player=17, Dealer=7, Ace=1	S	H	19.48%	41.67%	66.12706494692519	13.602266517209076	20.27066853586574	77
Player=17, Dealer=8, Ace=1	S	H	26.76%	44.90%	57.62412578328049	26.239526813771235	16.136347402948275	71
Player=18, Dealer=1, Ace=1	S	H	18.42%	28.89%	52.870272410300586	25.70578775512503	21.42393983457438	76
Player=18, Dealer=9, Ace=1	S	H	31.71%	54.17%	55.57544205318768	29.90801961728736	14.51653832952497	82

Figure 6: Model's decision through every state, usable ace is present.

For the states with no usable ace and where the agent and standard model differ, the agent performs better in 12/12 scenarios with a hand less than or equal to 14 (Figure 5). The standard model performs better in 6/6 scenarios with a hand greater than or equal to 15 (Figure 5). In states with a usable ace and where the agent and standard model differ, the standard model outperforms the agent model 8/10 times (Figure 6). So, one can question if the agent and standard model differ in 28 states, and the agent and standard model split the states in which they perform better, 14 each, why is the standard model's winning percentage about 2 percent higher? One reason as seen from the table is that one specific scenario in which the agent loses, non-usable ace hand of 16 vs a dealer 10 value, has 2000 counts - which means it appears far more than any other scenario. This state, 16 for a player's hand vs a dealer 10, is a very common scenario, especially compared to all the other hands. Mathematically it makes sense, as in the non-ace hands each scenario has about 450-490 counts, so multiplied by four for the four values

10 could be (ten, jack, queen, and king) the count is around 2000.

To generalize, our agent performs better from the standard model in situations where the user hand is in between the values 12-14 for non-ace situations, as it decides to hit and play more aggressively. Yet, the agent under performs at higher user cards and plays too passive (15+ player hand values both for non-ace and ace scenarios). The idea that in the states where the two models differed, the more aggressive strategy often performed better, made us question why this was the case. To find out more, we looked at the SHAP values for these states and the win rates for when we hit and stood.

5.2 SHAP Focus on Initial State Space

We first wanted to see where our model focused on initial states. This means looking at our SHAP values given every initial assortment of player's hand, dealer cards, and usable ace. One important clarification to note beforehand is that a high SHAP value doesn't always mean that a feature is of high value or even present. A high focus on the dealer's card could be because the card is very low and therefore of less use to the dealer, and a high focus on a usable ace feature could be due to the absence of a usable ace. SHAP values simply quantify a feature's contribution to the model's prediction, indicating how much that feature influenced the decision. With this in mind, Figure 7 depicts the results we found on initial hands.

Looking at Figure 7, there are a few pertinent takeaways from these results, the first being the focus on usable aces. It can be seen that for high middle (15-18) values and a low card for the dealer, the usable ace is a high focus point for the model when no usable ace is present. Compare this to the heatmap of SHAP focus when a usable ace is present, the usable ace is barely a high focus point and only contributes over 50% in one case. This indicates that the absence of a usable ace contributes a much larger focus to our model than a present usable ace.

The second key takeaway is how each feature contributes when the player's hand and the dealer's card are both low. The focus on the dealer's card is drastically higher than any other feature, as high as 92%, a degree of polarization barely seen anywhere else on the heatmap. This indicated that the model leveraged a weakness in the player's hand with a weak dealer card, and this intense focus comes up again later when looking at the model's action.

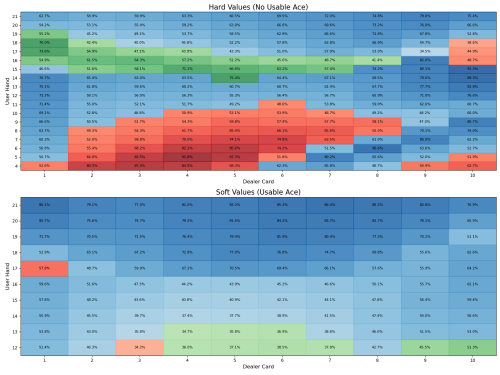

Figure 7: The most prominent SHAP focus for each combination of player hands and dealer cards, looking at hands with usable aces and non usable aces. Key: Blue is player's hand, Red is dealer's card, Green is usable ace

A final significant takeaway from these results are the outlier cases, which provided insight into how the model was "thinking." One outlier region is the focus on the dealer's 10-card when the player has a 16-18. This dealer card focus, though moderate and never above 50%, remains a larger contribution than the player's hand, a feature we might anticipate to be predominant in this context. We considered these values to be high risk (to hit on), compared to values such as 12-15 where the decision is less clear. We will see later that the high contribution of the dealer card here affects the model's decision to stand, but not when it decides to hit in these specific states, telling us the model considered it perhaps too risky. Looking at a player's 12-15 in this case, we see quite the opposite, with some of the largest focus on the player's hand on the entire map (over 90% with a dealer visible card of 10 and player hand of 15), a trend that carries through on the models decision to hit, emphasizing that the model found success through risk. The

other notable outlier here is the case where the user has a 17, a usable ace, and the dealer has an ace. The SHAP value shows almost a 59% contribution of the dealer card. The issue with a dealer having an ace as their upcard is that there is about a 33% chance the dealer will have Blackjack, and it should be noted that when the dealer has a 1 and the player a 17, the model never hits. This makes 17 a "threshold" value, where the dealer's odds of Blackjack doesn't enable the model to risk a bust, whereas a player's hand less than or greater than 17 provides a more clear decision for the model. When the player's hand is below 17 (shown later in Figure 9) the model will sometimes hit, reinforcing this assertion that 17 acts as a threshold.

5.3 SHAP Focus When Taking Action

It's not enough to just see where the model focuses when shown a hand, but how this focus transfers to its decisions. Figure 8 compares heatmaps that show how each feature contributed to the models decision on initial values in cases where it decided to hit and cases when it decided to stand.

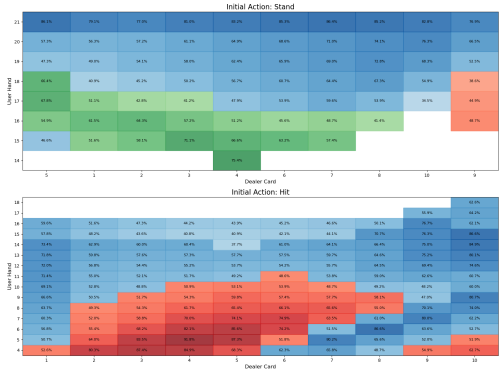

Figure 8: SHAP's focus on initial hands where the model chose to stand vs the model chose to hit. Key: Blue is player's hand, Red is dealer's card, Green is usable ace

Looking at Figure 8, both of the previously mentioned outlier cases disappear when the model chooses to hit (bottom heatmap), instead following more of an "axis" of focus where a low dealer card and a low player hand again bring a high contribution of the dealer's card, and inversely when both are high there is a high contribution of the player's hand. This diagonal axis of focus features no prominent contribution of the usable ace, and this absence of high focus on the ace is a glaring and conspicuous outcome that tells us the ace (or lack thereof) plays a much larger role in the action to stand, seen in the top heatmap.

At a high level, this means the model generally focuses on the usable ace and the player's hand when deciding to stand, and when choosing to hit it's due to the high contribution of the player's hand and dealer's card, especially around this axis of focus. This explanation is great as a general summary, but we wanted to try and understand why exactly the model took its action based on these values of contribution, to again try and make sense of what it was "thinking". Looking more closely at the results in Figure 8, player hand values of 14-16, especially when the dealer's card value were mid-range (the region where the heatmap for standing is heavily green and the heatmap for hitting is very light blue), no feature can really push the model's decision in one direction or another (a lack of an ace just means the user can't mitigate risk), constituting this as more of a "toss-up" region. The most prominent focus when hitting in this region is on the player's hand, but it's mostly in the low 40% range. We knew if the model decided to stand it's purely on a learned lack of risk in this region, where it can simply say the absence of a usable ace was its focus when doing so. This implies a usable ace is the only reason to hit here, a hypothesis that is later shown in Figures 10 and 11. Looking further, we wanted to see how our model acts when faced with these values, specifically the hit rate (Figure 9).

Looking at Figure 9, the region of variability is when a player's hand range from 14-16. In Figure 3, the model hits on 14 with every dealer's card except 5, and on 15 it stands with everything except a dealer card of 8, 9 or 10. Looking back at Figure 8, it can be seen that these are the exact regions where the SHAP focus is not particularly high on any feature. Since our model would choose to stand on a dealer 5 and

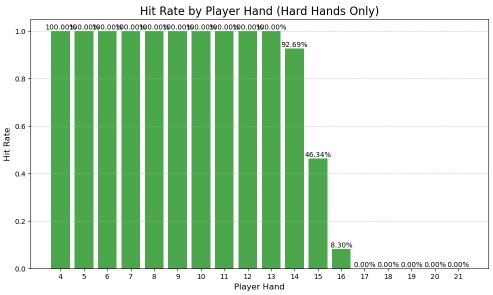

Figure 9: Hit rate of the model given a player's hand, with no usable ace

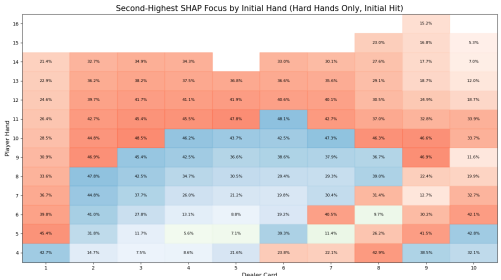

Figure 11: Second highest SHAP focus for every combination and the action is to hit, this time only with hard values

a player's hard 14, it can be deduced that this entire region of evenly dispersed (low) focus is where there is a present usable ace, and its presence is the sole reason why the model decided to hit. It's notable that even though the ace is why the model chose to hit in this region, it is not seen as the largest contributor. This again indicates toward the unexpected outcome that a usable ace, although present and a vital piece to the models decision, still isn't as big of a SHAP contribution as an unusable ace is. Looking further into the aces role in the decision to hit, Figures 10 and 11 show the greatest and second greatest SHAP focus on a hit action only when the usable ace is absent.

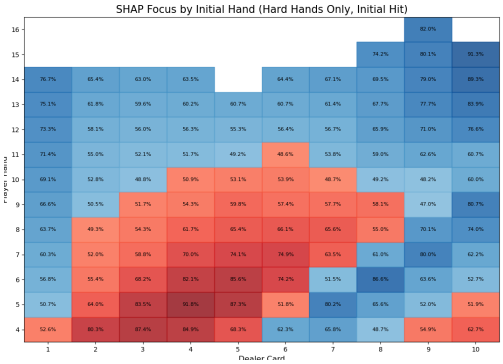

Figure 10: SHAP focus for every combination and the action is to hit, this time only with hard values

We see in Figures 10 and 11, the usable ace feature is virtually non-existent in the decision to hit. It can also be seen that the model now rarely hits on 15 or 16 here since no ace is present, following the action space shown in Figure 3. This indicates the action to stand in this region is solely rooted in the absence of an ace.

5.4 Back to the Research Question in Terms of Action

Piecing it all together, part of the research question explores how a Blackjack model can explain its reasoning for human understanding. Given a state space, how does the model decide to take action and can it be derived? We found that in the action space, when the model decides to hit, its focus shifts primarily to the player's hand and the dealer's card, with little to no emphasis on the usable ace. This highlights a polarized decision-making strategy based on these two features, shown most notably with the apparent diagonal axis of focus. Conversely, when deciding to stand, the focus often includes the presence of a usable ace, reflecting the model's risk-aversion in certain scenarios. In cases where no single feature strongly influences the decision, the model appears to rely on learned patterns of risk mitigation, further emphasizing this role of a usable ace as a critical factor in its reasoning. Looking at these results allowed us to determine the logical decision-making process of our model and its ability to reason when performing an action. The analysis here provided us some better insight into how our model was "thinking" when making decisions and how it can be used to inform a human on Blackjack strategy.

6 Future Work

One point for future work is to include memory of the last few states, to see if the agent can create card counting strategies, possibly even simulating how it creates this strategy with various deck sizes. For example, with lower amounts of decks used, card counting has much more of an impact. This makes it interesting to see how much the agent considers the counting card strategy in its decision-making process as the number of decks increases.

Another possible next step would be to further expand the state space. An example could be to represent a user's hands as totals and incorporate information about each individual card to allow for strategies like splitting the cards. Another logical place to expand is adding more actions the agent could do. Adding betting, more detailed representations of cards, and even other players could create better simulations of casino Blackjack play. This would mean diverging away from OpenAI's gymnasium state space and creating a new, more complex state space, which would also be a new aspect of the project.

Further studies could also look at different architectures for the neural network. Researchers could keep testing variations of the multi-layer perceptrons, trying out different numbers of layers, neurons, and activation functions to see new relationships. Lastly, research could be done to investigate and test different Q-learning algorithms, like double Q-learning algorithms, and integrate more explainability techniques, such as LIME.

7 Summary

In this research, we show it is possible to create a Blackjack agent capable of playing near the optimal win rate while also providing insights into its decision making process. We specifically used the SVG-DQN algorithm with a multilayer perceptron over the gymnasium Blackjack state space to achieve a win rate of 42 percent. By utilizing the Shapley-Shubik algorithm, we quantified the contributions of each feature (player's hand, dealer's card, and usable ace) to the agent's decision to hit or stand. Our findings showed the agent's divergence from standard play was due to more aggressive play for lower hands and more passive for higher hands, leading to performance differences between the two strategies. Although the agent performed slightly worse than the standard model, it was overall still able to perform better in half of the states it diverged from the standard model. Next, the model's SHAP contributions reveal a strong emphasis on the absence of a usable ace, particularly in mid-range player hands, and a focus on the dealer's card when both hands are weak. While hitting decisions prioritize the player's hand and dealer's card, standing decisions incorporate the presence of a usable ace, reflecting the model's polarized strategies and risk mitigation patterns.

Overall, the project shows the potential for an agent to perform complex probabilistic tasks at a high level while being able to explain the rationale behind the strategy it employs. Hopefully, explainable agents could become powerful enough to be used in game-based learning environments and for decision-making under uncertainty. Future work could focus on extending our methods to more complex environments.

References

[1] Blackjack Apprenticeship. Blackjack strategy charts - how to play perfect blackjack. Online resource. Accessed: 20 September 2024.

[2] Thomas Hickling, Abdelhafid Zenati, Nabil Aouf, and Phillippa Spencer. Explainability in deep reinforcement learning, a review into current methods and applications. *arXiv preprint (if applicable) or specify a journal if known*, 2023. 1, 1 (March 2023), 32 pages.

[3] Mohammed Sabry Khartoum and Amr M. A. Khalifa. On the reduction of variance and overestimation of deep q-learning. *arXiv preprint arXiv:1910.05983v2*, 2024. Accessed: 14 April 2024.

[4] Volodymyr Mnih. Playing atari with deep reinforcement learning. Technical report, University of Toronto, 2024. Accessed: 4 November 2024.

[5] Yunpeng Qing. A survey on explainable reinforcement learning: Concepts, algorithms, and challenges. *arXiv preprint arXiv:2211.06665*, 2023. Accessed: 1 November 2023.

[6] Web Services, Amazon. Interpretability versus explainability - model explainability with aws artificial intelligence and machine learning solutions. Online resource. Accessed: 21 September 2024.

[7] WinStar World Casino and Resort. Blackjack odds – know the odds of winning. Online resource. Accessed: 7 November 2024.

[8] Allen Wu. Playing blackjack with deep q-learning. Technical report, Stanford University, 2018. Accessed: 2018.

[9] Wei-Ye Zhao. Stochastic variance reduction for deep q-learning. *arXiv preprint arXiv:1905.08152*, 2019. Accessed: 20 May 2019.