Evaluating AI Agents to Solve the Blackjack Problem

Jake Seracino, Gabriel Camilleri, Jacob Cassar Ellis, David Vella

ABSTRACT

Blackjack is one of the most popular casino games in the world. It involves comparing cards between the players and the dealer. In this research, we implemented a number of AI agents adapted from several machine learning techniques that could solve the Blackjack problem. Each algorithm is designed to approximate the most optimal strategy which dictates what action should be taken given a particular game state so as to maximise winning likelihood. The three algorithms implemented are Q-Learning, an evolutionary algorithm and an evolutionary neural network. Whereas typical studies conducted in the domain focus mainly on three legal actions; hitting, standing and doubling-down, our contribution also considers splitting as this action is allowed in most casino variations of Blackjack. The aforementioned algorithms are also compared against each other to see which one performs best. Finally, the knowledge learnt by the AI agents was transferred into a Unity-based Blackjack simulation so as to allow the user to see in real-time the decisions taken by the agent given a particular game state.

KEYWORDS

Reinforcement Learning, Neural Networks, Evolutionary Algorithms

1 INTRODUCTION

Blackjack, or Twenty-One as it is sometimes referred to as, is a common casino game which involves comparing cards between players and a dealer [9]. The game is a casino banked game where players compete against the house instead of against each other.

The goal of the game is to get a set of cards also known as a hand, with a total value of 21 or closer to 21, such that the player get a total higher than the dealer's hand. To calculate the value of a given hand, each card is counted at face value with picture cards valued as 10s. The only exception is the ace, which is valued as either an 11 or a 1 at the player's discretion. A hand which contains an ace which is being valued as an 11 is known as a soft hand whereas all other hands are known to be hard. The player loses the round if the value of the cards in his hand exceeds 21. Losing in this manner is known as busting. Similarly, if the player has not busted and the dealer's hand exceeds 21, the player wins the round.

The game itself is played with multiple decks of cards, each comprised of 52 distinct cards. Typically, the number of decks in a casino game ranges between 4 and 8 [12]. Upon the start of a round, after each player places his original bet, the dealer draws two cards for each player as well as two for himself, but reveals only one of these cards. This card is known as the up card. Each player then proceeds to perform any of the legal moves in the game:

- **Hit** – prompts the dealer to give a new card to the player;
- **Stand** – ends the turn of the player so that the house can play its turn;
- **Double-Down** – adds an amount equal to the original bet to the bet, performs a hit and afterwards stands; and
- **Split** – splits the current hand into two distinct hands and adds an amount equal to his bet on the original hand to act as the bet on the second hand.

Each of the latter two moves is only allowed at the start of a round, that is, before the player performs a hit. Splitting also requires the player to have a pair of the same face cards. Other variations of blackjack may also offer surrender and insurance, but for the purposes of this study, these will not be considered.

If the player at any point busts, his turn will end there and he will lose his original wager. On the other hand, if the player opts to stand he will forgo the right to perform any other action and his turn ends. After all players have stood or busted, the dealer reveals the card which he had left concealed—known as the down card—and plays his turn. At any given point, the dealer may only stand or hit. The actions he performs are constrained by the variation of the rules of the blackjack game being played.

Typically, the dealer is forced to hit until his hand is valued at 17. Another common variation is one where the dealer also hits on a soft-17 while standing on a hard one. Given that this variation has seldom been studied in literature, we will be conducting this research using this variation.

Whilst hobbyists may play blackjack based on feeling or more colloquially, gut instinct, skilled players have developed techniques and strategies which maximise their likelihood of winning. This is possible because the game is stochastic in nature and, as such, can be analysed statistically [7]. It is therefore interesting to try to apply machine learning techniques to conclude if such algorithms can *learn* the game of blackjack to the extent of being able to consistently win and minimise house edge.

2 BACKGROUND RESEARCH

2.1 Statistical Analysis of Blackjack

The fact that blackjack is a stochastic problem, that is, it has a random probability distribution, means that it can be analysed and studied in a manner similar to other statistics problems. Baldwin et al. in [2] mathematically derived the most optimal strategy for blackjack. This strategy dictates what is the best course of action the player can take given the card he has in his hand and the dealer's up card. This study has served as a basis for all future studies of the game of blackjack and remains relevant today since modern strategies all derive from it.

One of the most important publications with regards to the game of blackjack is [15] by Thorp. This book expands on the strategy proposed by Baldwin et al. [2] to cover more variations including in the number of packs being used as part of the game. This strategy is known as Basic Strategy and is the one which is found most commonly around the world [9]. Interestingly, Thorp also proposes a separate strategy for expecting what cards are remaining in the deck so as to have an idea of what cards should be drawn next. This strategy is known as Thorp's Ten Count. At an abstract level, this technique works by maintaining a ratio of non-ten cards and ten cards. Alternative methods used for this prediction exist with the most common of which being HiLo [16]. These strategies are known as card counting. When card counting is used, dynamic betting strategies can be applied so as to improve winnings since 'knowing' what cards are left in the deck continues to diminish the house advantage. Although such techniques can lead to a more robust blackjack agent, integration of card counting systems will be left for a future study.

2.2 Q-Learning

Q-Learning is a form of model-free reinforcement learning based upon temporal difference [17]. At each time step, given a particular state, the agent tries an action and evaluates its immediate consequence. When the consequence is positive, a reward is given whereas when the consequence is negative, a penalty is applied. The equation that is used is given by:

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma \times max_a(Q^*(s',a')) - Q(s,a))$$

where $Q(s,a)$ refers to the value of the action a given state s, r refers to the reward given during this iteration, α is the learning rate, γ is the discount factor and $max_a(Q^*(s',a'))$ refers to the value of the highest-valued action given the next state. One other important aspect of Q-Learning algorithms is the exploration rate ϵ. This is the rate at which the agent decides to not take the greedy approach—the action which has the greatest value given the current state—but explores other actions.

In literature, this technique has been previously applied to the blackjack domain such as in [7]. Here, de Granville modelled the state to take the form of a tuple formed by the total value of the hand of the agent, a boolean denoting if the hand is soft or hard, another boolean denoting if the hand may split and the value of the dealer's up card. Moreover, he allowed the same actions being considered for this study; hit, stand, double-down and split. On the other hand, he played the variation which forces the dealer to stand on a soft-17. Furthermore, he made use of 6 decks as well as paid a natural blackjack in the ratio of 3:2. In this study, de Granville managed to show that the Q-Learning algorithm was able to converge asymptotically to the performance of an agent making using of basic strategy but was not able to approximate it exactly.

2.3 Evolutionary Algorithms

Evolutionary algorithms are algorithms which aim model the collective learning process within a given population [1]. Each individual in the population is a single point in the solution space. The starting population is initialised using an algorithm-specific method. It is important to have a diverse starting population so as to avoid premature convergence to a sub-optimal solution [3]. This population is then evolved through the processes of:

- **Selection** – the process of selecting individuals from the population for breeding;
- **Crossover** – also known as recombination, is the process of combining the genetic information of two parents to generate an offspring; and
- **Mutation** – the process of altering genes of a given individual so as to maintain genetic diversity.

Selection of an individual is typically conditioned on the fitness—calculated using a problem-specific fitness function—of said individual. This is done because having fit parents drives individuals to be better and fitter themselves since an individual is formed through the crossover of his parent genomes. There exist a number of selection strategies which can be employed such as roulette wheel selection, rank selection and tournament selection, each with their own set of advantages and disadvantages to be considered. Examples of evolutionary algorithms include genetic algorithms, genetic programming and evolution strategy.

Cavarlee and Koza [4] came up with a standard genetic algorithm approach to overcome hardware limitations and approximate the most optimal strategy for blackjack. In their implementation, a gene of an individual in the population is, used to represent a decision given a particular state. The individual's genetic structure was therefore formed by a grid of h rows, representing all of the possible hands a player can be dealt, and w columns which represent all of the possible up-cards a dealer can have. In such a way each individual may be seen as a candidate strategy for blackjack. Through evolution, these candidate strategies approached the known optimum of the Basic Strategy but were unable to match the returns of it.

Other implementations that use evolutionary algorithms include the work of [5]. In this paper, the authors proposed a hybrid genetic algorithm which makes use of an iterative search method. It has been described as resembling hill climbing such that k is the number of simultaneous player hands the heuristic optimizes using a genetic algorithm in the inner most loop. The algorithm also makes use of spectral noise whereby, apart from depending on the hand of the player and the value of the dealer's up-card, the execution is also dependent on a noise fluctuation model. This creates a situation where the implied decision is a random variable.

2.4 Evolutionary Neural Network Algorithms

"EA is a random search algorithm which simulates the nature selection and evolution process (Xie 1997)."[8]. It is capable of performing a global search for the most optimal result without having to perform gradient descent. It makes use of evolutionary techniques, where it picks out the fittest individuals represented by a neural network, and maximizes their parameters to reach the most optimal solution to a problem. "Both Evolutionary algorithms and ANNs are the theoretical results of applying biological principles to the science research. In recent years, more and more researchers try to combine EAs with ANNs (Yao et al. 2004), hope that by combining the advantages of them, they can create a more efficient method"[8].

In [11] it is explained that defining the right parameters for a neural network is time consuming. Evolutionary neural networks are capable of learning what are the right hyperparameters to produce the fittest off-springs. Therefore, with a combination of evolutionary algorithms and deep neural networks, one can modify a neural network's hyperparameters to achieve state-of-the-art results. Previous research was done using a similar algorithm by Graham Kendall and Craig Smith in [10]. However, the approach taken was more advanced since they used three neural networks "to learn the game of blackjack. One network deals with the possibility of splitting, the second deals with doubling down, and the third deals with normal game playing."[10]. The approach taken in our research employs the neural network as though it was the player of the game since it chooses which action to perform in a given situation.

3 IMPLEMENTATION DETAILS

An important aspect for both training as well as evaluation is that of ensuring a fair setup whereby no model is favoured on another. In a regular blackjack game, the number of cards is limited by the number of decks that are in play and that the same physical card cannot be inside the hand of two or more players. Moreover, an unlikely but entirely possible situation may crop up where a particular player gets more favorable hands whereas another might get harder ones. For this reason, we created a function which allows N players to play blackjack under the same conditions. This means that each player draws the same cards and in the same order. Therefore, if the model wins, it will be on the merit of the actions it took and not on circumstance. The following list describes the rules which were adhered to when creating the blackjack version used in this study:

1. 8 decks, shuffled every 50 rounds;
2. Dealer stands on hard-17+ and soft-18+;
3. Double down allowed after splitting;
4. No re-split limit;
5. No surrender nor insurance; and
6. Natural blackjack pays 1:1.

3.1 Q-Learning

3.1.1 Overview

Our implementation of the Q-Learning algorithm is based on that proposed in [7]. We adopt the same state structure of having a tuple consisting of:

1. The value of player's hand;
2. A boolean denoting if the hand is soft;
3. A boolean denoting if the hand can be split; and
4. The value of the dealer's up card.

Nevertheless, some aspects of our implementation needed to be custom designed due to a lack of detail in the aforementioned paper. Namely, the manner by which splits' value is updated. The reason for this being that, essentially, performing a split leaves the player in two different states since this action leaves him with two distinct hands. Moreover, both of these hands' rewards should be used to revalue the split.

For this reason, we implemented a system whereby each state-action pair, hereby denoted as a *decision*, is stored in a list, denoted as an *episode*. The agent also maintains a list of these episodes. Whenever a split occurs, two new episodes are created consisting of all of the decisions taken up till that point in the original episode with the latter being removed. Moreover, each episode is relevant for a single hand. In such a way, the player is allowed to split an infinite amount of times with no caps necessary. After the turn of the player ends, we apply the value function to reward each decision in every episode. In such a manner, a split's value is amended for both of the states proceeding it but, since each episode is distinct from the other, for each revaluation, we are treating each state as if it were the only one. One might argue that we are modifying the Q-Learning algorithm, by giving the reward at the end rather than at every time step. Nevertheless, the function is applied in the exact same manner and the revaluation function only knows the next state just as with the regular application of Q-Learning.

3.1.2 Training

The training setup we employed makes use of the aforementioned play under same conditions function. Here, a number of Q-Learning agents are trained simultaneously by playing the game in the same environment. The agents trained are those formed by every combination of the following set of hyperparameters: $\alpha \in \{0.1, 0.01, 0.001\}$, $\gamma \in \{0.7, 0.8, 0.9\}$ and $\epsilon \in \{0.05, 0.1, 0.2\}$.

The actual training is performed over 1,000,000 rounds of blackjack. This number was chosen as it ensures that the vast majority of hand combinations are encountered by the agent learning to play the game, including less likely hands such as pairs. Finally, the betting strategy used by the agent is constant. At the start of every round, the agent bets 1 chip. If the player loses, the reward in the Q-Learning value function is taken to be -1, if he draws it is taken to be 0, if he wins its taken to be 1 and, finally, if he wins after doubling-down, the reward is valued as 2.

3.2 Genetic Algorithm

3.2.1 Overview

The architecture for our genetic approach to learn blackjack strategy is fundamentally identical to that found in [4]. We apply the same concept of modelling a single individual in the population as a strategy whereby each row represents the hand the player has been dealt whereas each column represents the value of the dealer's up-card.

3.2.2 Training

Training the genetic algorithm is a very expensive endeavour. The reason for this being that the calculation of the fitness function requires the population to play a significant number of rounds of blackjack, with the more rounds being played, the less variance in the fitness results. Just as with Q-Learning, each individual's fitness is calculated by having the entire population play simultaneously under the play under same conditions function. This ensures that we are able to compare directly the performance of each individual with the others and eliminates luck from the equation. It was decided that each generation plays 10,000 rounds of blackjack.

The fitness function itself is rather simple as we made use of the net chips at the end of the 10,000 rounds. This heuristic made sense as the goal of all casino-goers is to try and make as much money as possible, therefore the algorithm with the most money at the end is the one which performed best. It is good to note that all algorithms will end up with a negative number of chips. We know this because Basic Strategy—the mathematically proven best strategy for blackjack—yields negative returns [5] and as such, even if the genetic algorithm manages to converge to Basic Strategy, the net returns will be negative.

Each algorithm was trained with a single combination of the following hyperparameters:

- **Population Size** – 200, 300, 400
- **Mutation Rate** – 0.005, 0.01
- **Selection Method** – Rank, Tournament
- **Tournament Size** – 3, 5 (Relevant only when selection method is set to Tournament)

Finally, elitism was not considered for this implementation.

3.3 Evolutionary Neural Network

3.3.1 Overview

Deep Neural Networks and Genetic Algorithms separately have common uses in attempting to solve real world problems, as well as games such a blackjack. However, "DNNs depends on the proper configuration of its architecture and hyperparameters. Such a configuration is difficult and as a result, DNNs are often not used to their full potential."[11] Choosing the right deep neural network architecture is mostly a case of trial and error, where the hyperparameters are fixed to verify if the result improves or not. Configuration is therefore based on history of the previous DNN configurations. The evolutionary neural network is a method to automate the process of choosing the best neural network with the right hyperparameters that guarantee a success. "Therefore, automated configuration of DNNs is a compelling approach for three reasons: (1) to find innovative configurations of DNNs that also perform well, (2) to find configurations that are small enough to be practical, and (3) to make it possible to find them without domain expertise."[11]

The architecture of our evolutionary neural network algorithm is simply made up of two key elements. A simplex neural network and a genetic algorithm. The simplex neural network is simply a three layered neural network with one hidden layer. The hidden layer size is by default set to 100. Each layer makes use of an activation function, in our case we made use of the sigmoid activation function. The generation size was also by default set to 20 and the mutation rate was set to 0.05.

The simplex neural network acts as one player. The neural network picks its own hyperparameters and is trained using backpropagation. Therefore, initially the genetic algorithms starts off with 20 neural networks or players in this case. The neural networks are then passed on to the genetic algorithm for further training, or in the case of genetic algorithms, mutations. For each network passed random values are added to some weight elements based on the mutation rate.

The network's performance is based on three measurements; the winning rate, the average amount left in the bank and the reward received. The cost is then calculated on these three variables. The cost is finally returned to reflect the performance of the neural network. The idea behind this, is that we want to maximise the likelihood of having a player that is capable of winning frequently, making best use of the amount at hand and able to make more money from the amount bet. Evolutionary Algorithms "EAs are another class of algorithms widely used for black-box optimization of complex, multimodal functions. They rely on biological inspired mechanisms to improve iteratively upon a population of candidate solutions to the objective function."[11].

The blackjack game architecture is based similar to the rest of the other algorithms developed. Each value the dealer has and the player has represents a state one is in. In each state is the action it takes to go to the next state. The transition values are represented by 'H' for hit, 'D' double-down, 'S' for stand, and 'P' for split. The transition table is filled up throughout the training process, by picking the players which won the round and taking their actions done into winning into consideration.

3.3.2 Training

The algorithm required further development to achieve optimal results. However, with enough training, a few optimal players are generated. Mutation is very unstable and does not follow a normal distributive pattern. One has to keep in mind that "the performance of neural networks is sensitive to the number of neurons. Too few neurons can result in poor approximation, while too many neurons may contribute to overfitting problems.”[8]. On the other hand, “EA is a random search algorithm which simulates the nature selection and evolution process (Xie 1997). It has the advantage of good global searching capability and learning the approximate optimal solution without the gradient information of the error functions”.[8].

The algorithm was fully trained to produce one complete state-transition tables on the following hyperparameters; mutation rate of a 0.01, generation size of 20, and using a maximum epochs of 30,000. Each player played 5 rounds each, and depending on whether they won or not their actions were inputted into the table only once.

4 EVALUATION

4.1 Q-Learning

Our evaluation strategy for the Q-Learning algorithm was to deduce three candidate hyperparameter solutions which performed best under these three ordering criterion:

1. The combination which won the most rounds from the testing rounds;
2. The combination which lost least. This is not the inverse of winning since in blackjack, the player may get a draw with the dealer; and
3. The combination which had the best amount of net chips after the end of the testing rounds.

These criteria were chosen because they made the most sense. For example, winning the most amount of rounds ensures that agent is consistently beating the dealer whereas minimizing the number of rounds in which the player loses means that the agent will minimize the number of times in which he loses his original wager. Finally, the net amount of chips is considered because at the end of the day, every casino-goer’s goal is to try to win as much money as possible while playing. So as to ensure that during testing the agents experience as wide a spread of states as possible, we set the amount of testing rounds to be 100,000. Once more, the algorithms were tested under the same environment using the play under same conditions functions so as to ensure fairness in out tests.

Initial experiments performed on the 27 agents trained showed that the three top-performing agents under each of the aforementioned criteria were:

- **Most Wins** – α: 0.01, γ: 0.8, ϵ: 0.2
- **Least Losses** – α: 0.01, γ: 0.8, ϵ: 0.05
- **Best Net Chips** – α: 0.01, γ: 0.7, ϵ: 0.2

Interestingly, the best solution for each criterion was different for the three of them. This means that the agent which won the most games did not have the best net stack of chips. Similarly, the agent which lost least did neither win most nor have the best stack of chips at the end. Table 1 below shows the results for these three agents.

α	γ	ϵ	**Wins**	**Losses**	**Chips**
...					
0.01	0.7	0.2	44643	49828	-6076
0.01	0.8	0.05	44535	49463	-6834
0.01	0.8	0.2	44681	49866	-6916
...					

Table 1: Excerpt from the table showing the number of wins, number of losses and the net amount of chips after 100,000 testing rounds of blackjack for all of the trained agents.

For this reason, we performed again the test for these three agents under the same setup. The graphs from this experiment are shown in Figure 1 and Table 2 shows its results.

α	γ	ϵ	**Wins**	**Losses**	**Chips**
0.01	0.7	0.2	44821	49742	-5997
0.01	0.8	0.05	44630	49335	-6458
0.01	0.8	0.2	44781	49810	-6733

Table 2: Table showing the number of wins, number of losses and the net amount of chips after 100,000 testing rounds of blackjack for three agents.

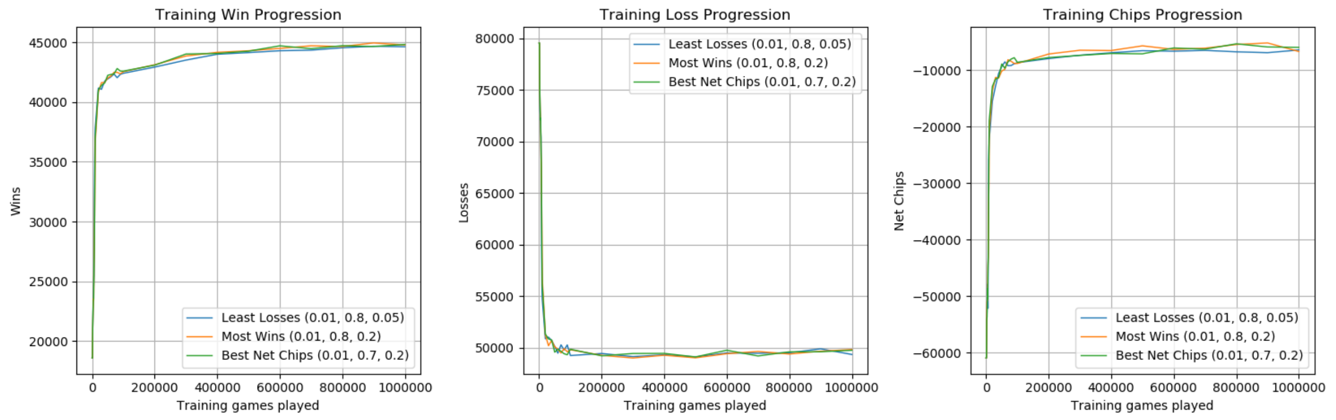


Figure 1: The graphs showing how the number of wins increased, how the number of losses decreased, and how the net amount of chips changed given a number of training hands.

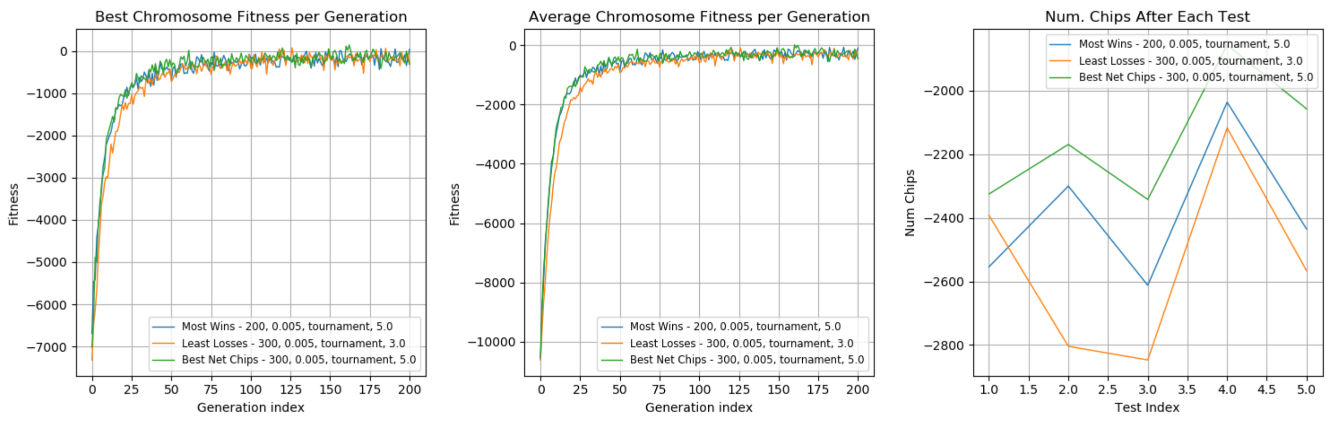


Figure 2: The graphs showing the best chromosome fitness of each generation of the top performing genetic algorithms, their average generational fitness.

It can be seen that the agent which lost least in the first test did so again in the second one. On the other hand, the agent which had the won the most rounds in the first experiment was overtaken by the agent which had the best net amount of chips. The latter also maintained the top spot in this criterion. On the other hand, the graphs in Figure 1 show that the difference between each agent is negligible at best but nevertheless, we will consider the hyperparameters α: 0.01, γ: 0.7, ϵ: 0.2 to be the optimal combination for our Q-Learning agent.

4.2 Genetic Algorithms

The process to evaluate the genetic algorithm is a rather simple one. We performed five consecutive tests of 100,000 rounds so as to assurance a good level of confidence in our results. Similar to our experiment conducted on the Q-Learning agents, we recorded the number of wins, number of losses and the net amount of chips after each test. The top three performing algorithms in each of these criteria were:

- **Most Wins** – Pop. Size: 200, Mut. Rate: 0.005, Sel. Method: Tournament, Tour. Size: 5 ... Denoted **A**
- **Least Losses** – Pop. Size: 300, Mut. Rate: 0.005, Sel. Method: Tournament, Tour. Size: 3 ... Denoted **B**
- **Best Net Chips** – Pop. Size: 300, Mut. Rate: 0.005, Sel. Method: Tournament, Tour. Size: 5 ... Denoted **C**

Just as with Q-Learning, all of the three criteria had different top performers. We can also see that the hyperparameters of the selection method and the mutation rate are common throughout; Tournament and 0.005, respectively. This shows that both of these hyperparameters are the best for this task from those that we tested. The three algorithms above's fitness improvement over all generations as well as their performance with regards to net chips is given in Figure 2.

G.A.	μ_w	σ_w	μ_l	σ_l	μ_c	σ_c
...						
A	43950	76.0	48536.6	133.4	-2387.4	205.7
B	43875.8	58.6	48484.4	112.9	-2545.2	270.3
C	43949.8	67.6	48663.0	131.1	-2149.8	180.5
...						

Table 3: Excerpt from the table showing the mean μ and standard deviation σ of wins (w), losses (l) and net amount of chips (c) after 5 iterations of 100,000 testing rounds of blackjack for each trained genetic algorithm.

Table 3 shows that genetic algorithm C won almost as much as algorithm A but the lower standard deviation increases our confidence in it more than the latter. Moreover, not only did it have the highest returns from all of the genetic algorithms, but it also had a comparatively lower standard deviation than all of them. For this reason, we were able to conclude that from all of the genetic algorithms we tested, C is the best one.

4.3 Evolutionary Neural Networks

The algorithm was tested with different hyperparameters with 5000 epochs each. Each algorithm has a generation size of 20, this can be increased for further testing. However, due to hardware limitation we set it to the most minimum, just enough to get results. The first test was with a generation size of 20 and a hidden layer size of 100. For the hidden layer a sigmoid function was used.

The second phase of testing was with different hyperparameters, so as to see how the result varies. The generation size was set to 50 with a hidden layer size of 300. The hidden layer used a Tanh activation function. It was duly noted that both tests showed a similar convergence. Initially the A.I. learns quickly then slows down to learn the more complicated actions. Both tests show that the algorithm needs further training, to reach its optimum. It is also worth noting that with little amount of training the A.I. stands at early stages. This approach was later changed, where the A.I. was then capable of changing previously visited states and modify them appropriately with more training at hand.

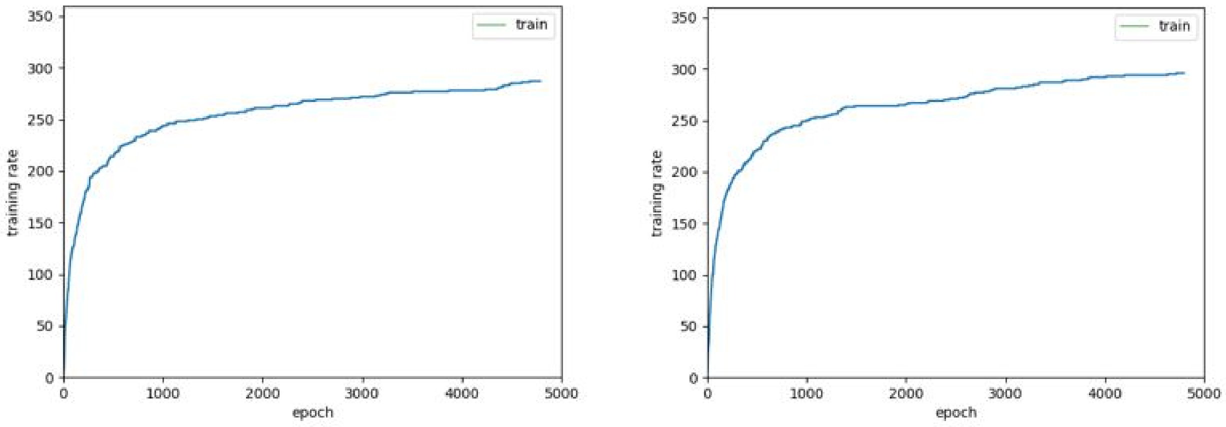


Figure 3: Learning rate of test 1 and test 2 of the evolutionary neural network.

4.4 Final Evaluation

After we selected the best trained agents of each algorithm, it was fitting to see how these compare to one another. Moreover, these were compared to an agent which was playing using Basic Strategy. The latter acted as our gold standard since Basic Strategy is known to be the optimal strategy for blackjack. The algorithms were also be compared to three baselines:

- **Mimic-the-Dealer** – This is a strategy whereby the agent plays using the same strategy as the dealer. In our case, this is hitting on all soft and hard hands up to a value of 16 and also on a soft-17;
- **Never-Bust** – With this strategy, the agent stands on all hands which have a value of greater than or equal to 12. As the name implies, with this strategy, the player can never bust; and
- **Random** – This baseline cannot be described as a strategy since the agent does not pick any particular action given a state. In fact, the agent simply plays by picking one of the legal moves for his hand at random.

The testing setup is similar to that employed for the genetic algorithms. We make use of 10 consecutive tests of 100,000 blackjack rounds which are played under the same conditions for each agent. The choice of having 10 tests was so as to recognise the presence of variance, if it were to be present, as well as to be able to perform some basic statistical analysis.

Figure 4 shows the results of this experiment. As expected, Basic Strategy yielded the best results in terms of the net chips the player had after playing with it. Still, it was outperformed in terms of total wins by Q-Learning and by the genetic algorithm in terms of least wins. Moreover, we can see that the amount of times Q-Learning lost was clearly greater than both Basic Strategy as well as the genetic algorithm. Considering the fact that the algorithm also had the most wins, we can deduce that the algorithm was more aggressive than the others in terms of style of play, that is, it looked for the win more than playing safe so as to avoid busting. This contrasts with the genetic algorithm which had clearly less wins but also less losses, showing a safer strategy. This safer strategy also shows that it yielded a better net amount

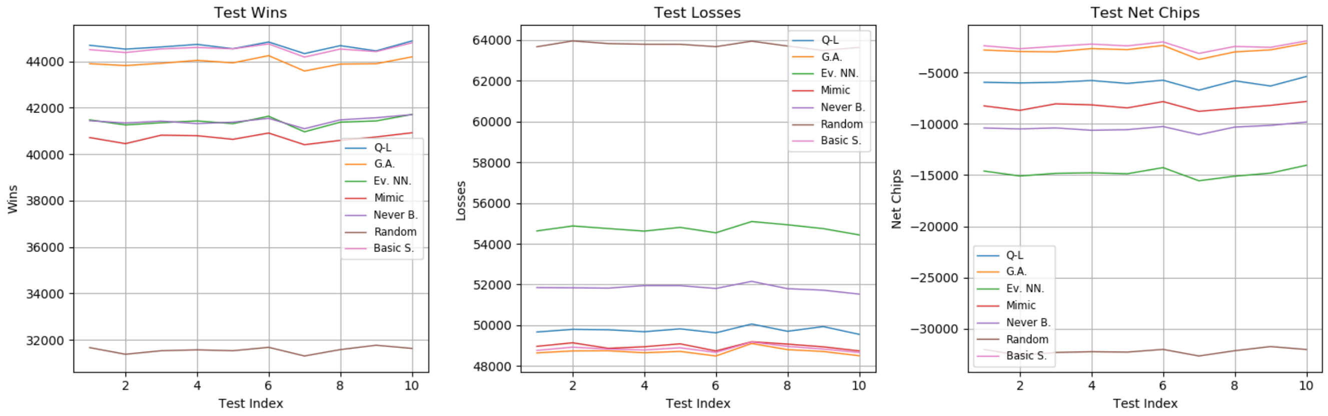


Figure 4: The graphs showing the comparison of the three algorithms with the Mimic-the-Dealer, Never-Bust and Random baselines as well as with the Basic Strategy gold standard.

of chips as it is much closer to Basic Strategy than Q-Learning. The evolutionary neural network was outperformed by both the Genetic Algorithm and the Q-learning algorithms in terms of wins, losses and net chips. The interesting thing is that it seems to perform similar to the Never-Bust algorithm, in terms of winning.

We also found that both Q-Learning and the genetic algorithm clearly outperform all of the baselines in terms of rounds won and the net amount of chips. Nevertheless, in terms of number of losses, the Mimic-the-Dealer baseline's performance is comparable to that of the genetic algorithm and even performs better than the Q-Learning algorithm. This is mainly due to the fact that the dealer plays in such a manner which reduces the chance that he busts. On the other hand, the evolutionary neural network's performance was only comparable to the Mimic-the-Dealer and Never-Bust baselines in terms of rounds won. In the other criteria its performance was poorer.

An interesting point was that although the number of fitness rounds for the genetic algorithm was relatively low (10,000) for each generation, the algorithm managed to converge almost to the optimal strategy. We hypothesise that this is due to the fact that we tested the population of each generation under the same conditions. This ensured that we eliminated the presence of variation and allowed us to directly compare each individual without the need of a greater amount of rounds.

Alg.	μ_w	σ_w	μ_l	σ_l	μ_c	σ_c
Q-L	44624.5	160.6	49764.4	142.7	-5975.4	338.0
G. A.	43936.8	177.9	48712.4	162.5	-2820.0	399.1
E.N.N.	41392.5	194.9	54742	184.9	-14798.4	405.1
Mimic	40696.4	167.3	48970.2	148.1	-8273.8	312.7
Nev. B.	41426.4	156.1	51842.2	153.4	-10415.8	306.4
Rand.	31560.9	131.5	63748.8	135.8	-32187.8	264.2
Bas. S.	44520.2	167.7	48851.1	143.8	-2422.6	324.7

Table 4: The table showing the mean (μ) and standard deviation (σ) of wins (w), losses (l) and net chips (c) from the experiment conducted to compare Q-Learning (Q-L), the Genetic Algorithm (G.A.) and the Evolutionary Neural Network (E.N.N.) with the Mimic-the-Dealer (Mimic), Never-Bust (Nev. B.) and Random Strategy (Rand.) baselines as well as with the Basic Strategy (Bas. S.) gold standard.

Table 4 shows mean and standard deviation for all of the statistics which produced the graphs in Figure 4; wins, losses and net chips. We can see that in all cases the tests produced very consistent results with a small standard deviation, compared to the number of rounds played which amounted to 100,000 for each test. This increases our confidence in the results from the tests and in our conclusions which are that the genetic algorithm approximated a strategy which yields better returns and is closer to Basic Strategy than Q-Learning and the evolutionary neural network.

5 UNITY SIMULATION

5.1 Overview

A simulation was built to allow the users to both play and see the artificial intelligence play the blackjack. The game was built using the Unity Game Engine as per the requirements of the assignment specification. The game is made up mainly of five scenes; the Main Menu, the Real Player scene, the Evolutionary Neural Network algorithm scene, the Q-learning algorithm scene, and the Genetic algorithm scene.

The Real Player scene allows the user to play against the dealer, whilst the Q-learning, the Genetic and the ENN are simulations for the algorithms to play against the dealer. The game makes use of 8 deck of cards, to mimic the environment in which the Artificial Intelligent algorithms were trained in. The game is playable on Windows.

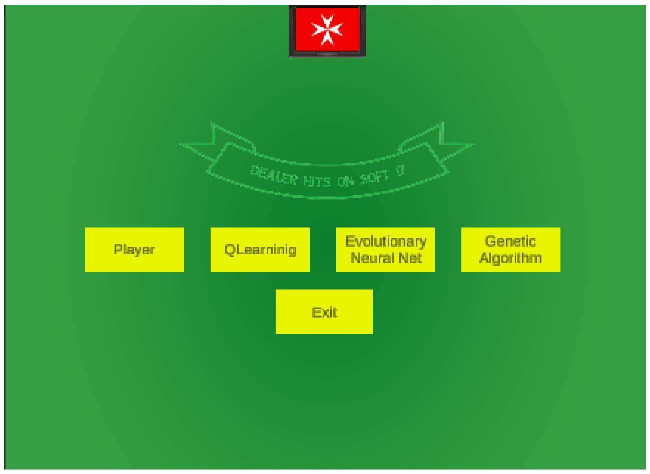


Figure 5: The Main Menu

5.2 Architecture

5.2.1 Main Menu

The main menu is made up of five button; the *Player* button, the *QLearning* button, the *Evolutionary Neural Net* button, and the *Genetic Algorithm* button, all of which leading the user to different blackjack game scene. Finally, the *Exit* button terminates the game.

5.2.2 Real Player

The architecture of the Real Player is made up of a series of buttons, which once triggered will execute a particular blackjack action. The user is provided with four main action buttons; *Double*, *Hit*, *Stand*, and *Split* button. Once clicked, they perform the associated blackjack action. The user is also provided with different chip icons whereby each icon represents a different amount to be added to the overall bet of the current round. Once a bet has been placed, the user is notified by how many chips he has remaining in his stack as well as how much in total has been bet.

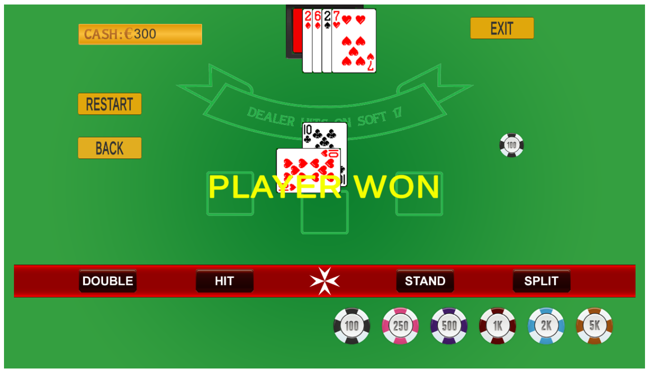


Figure 6: Game Scene

Finally, at the bottom of the screen, the user is provided with a text bar which must be used during a split. In it, the user must input which hand they wish to take an action on. When splitting, the new hand goes to left hand side of the current hand, due to screen size limitations. Therefore, when hitting or doubling-down on a particular hand, one can simply input the number of the hand with 0 being for the right most hand.

5.2.3 *Artificial Intelligence*

To connect the AI agents with the unity game, we decided to make use of the *Pulling lever* idea. In layman's terms, each agent acts as the mind which triggers an intended action by clicking the associated button accordingly. As discussed previously, each algorithm generates a blackjack strategy table of its own. The table is fed into the game through a CSV file and according to the dealer's Up-card value and the player's hand attributes and whether it is splittable or not. The actions within each strategy table is represented by a letter; *H* for hitting, *D* for doubling down, *P* for splitting, and *S* for standing. Depending on which state the game is in, the action dictated by the strategy is called and the respective button is clicked. In turn, this results in the right animations being triggered so as to take the game to the next state. Finally, the game specifies whether the dealer or the player won. During the game, the player is free to go back to the Main menu or restart the game.

5.3 Evaluation and Improvements

Overall the game plays smoothly. However, the AI agent simulations seem to play the animations all at once without allowing some seconds to pass between each animation. This problem is due to synchronous and asynchronous commands. Unity seems to read the code and actions all at once and place them in a Queue. Finally, the actions are all handled from the queue at once. Therefore, one improvement would be to allow the user to slowly observe each action taking place during the game, whilst the algorithm is simulating the game. Another limitation was screen size. We were highly limited by the number of animations and gameobjects which could be presented within the game, for example, the user is limited by how much they are able to split hands, even though the algorithms can split the hand more than twice although such a scenario is very unlikely due to the extremely low probability of successively splittable hands. An improvement to our current implementation, would be to allow the player to play against the dealer along with A.I. agents within the same environment.

6 FUTURE WORK

As an improvement to our current algorithms in solving blackjack, card counting can also be introduced. Card counting can determine whether the next hand is estimated to whether or not it is going to be of an advantage to the player or to the dealer. Many more algorithms can be used to solve the blackjack problem in the most optimal way. In [13] it is discussed that blackjack can be solved using the Q-SARSA algorithm where the algorithm learns a Markov Decision process policy to learn how to play blackjack. The algorithm gave "good results for the blackjack problem, but the results were no better than the results that could be received by the Q(λ) or the SARSA(λ) algorithms alone."[13].

Richard S. Sutton and Andrew G. Barto in [14] discuss the idea of solving the blackjack problem using the Monte Carlo prediction. They consider the idea where each player plays independently against the dealer. "To find the state value function for this policy by a Monte Carlo approach, one simulates many blackjack games using the policy and averages the returns following each state."[14]. One fact drawn about the Monte Carlo methods is that the estimates for each state are individual. Therefore one state does not build up its information from another state.

A third way how we could solve the blackjack problem is using Deep Q-Learning. "Traditional Q-learning is a powerful reinforcement learning algorithm for small state-action spaces, but performs poorly on more involved systems".[18].Given our intention was to solve the problem using the most complicated actions in order to win, deep Q-learning may help us in efficiently finding solutions to the most complicated action in the blackjack problem. In [18] an -greedy exploration with annealing was used, to improve exploration at the beginning of the training. It can also improve exploitation towards the end. We can also benefit from using an experience replay buffer as mentioned in [18], something which traditional Q-learning lacks, such that it "can underestimate the value of uncommon actions and overfits to recent experiences"[18].

Another approach which can also improve the results is to use a similar approach to [6]. Where "Populations of neural network agents evolve using genetic algorithms (population learning) and at each generation the best performing agents are selected as teachers."[6]. This method is known as Cultural Learning. It is the process of passing on information between generation without use of genetic material. Using their method along with the one in [10], can allow us to generate multiple simplex neural networks each with different hyperparameters, and the ones that are the fittest are chosen to teach the newly generated neural networks. Thus, the algorithm can be further improved with such modifications and features.

7 CONCLUSIONS

Throughout this study, we explored various machine learning techniques so as to determine their ability to be applied to the domain of learning the optimal strategy for blackjack. We presented a description about the game and referred to literature which shows that this problem can be learned using machine learning due to it being stochastic in nature. Moreover, we gave some background about the three algorithms which were applied to the domain; Q-Learning, Genetic Algorithms and Evolutionary Neural Networks, as well as how they have been previously applied to the domain. We also conducted some experiments on our implementations of these algorithms and presented the observed data. We were also able to conclude that the genetic algorithm implemented approximated a better strategy for blackjack than Q-Learning and the Evolutionary Neural Network.

REFERENCES

[1] BACK, T. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[2] BALDWIN, R. R., CANTEY, W. E., MAISEL, H., AND MCDERMOTT, J. P. The optimum strategy in blackjack. *Journal of the American Statistical Association 51*, 275 (1956), 429–439.

[3] BHATTACHARYA, M. Diversity handling in evolutionary landscape. In *International Workshop on Combinations of Intelligent Methods and Applications* (2014), Springer, pp. 1–8.

[4] CAVERLEE, J. B., AND KOZA, J. A genetic algorithm approach to discovering an optimal blackjack strategy. *Genetic Algorithms and Genetic Programming at Stanford* (2000), 70–79.

[5] COLEMAN, R., AND JOHNSON, M. A. Genetic algorithm-induced optimal blackjack strategies in noisy settings. In *Advances in Artificial Intelligence* (Berlin, Heidelberg, 2004), A. Y. Tawfik and S. D. Goodwin, Eds., Springer Berlin Heidelberg, pp. 467–474.

[6] CURRAN, D., AND O'RIORDAN, C. Evolving blackjack strategies using cultural learning in multi–agent systems.

[7] DE GRANVILLE, C. Applying reinforcement learning to blackjack using q-learning.

[8] DING, S., LI, H., SU, C., YU, J., AND JIN, F. Evolutionary artificial neural networks: a review. *Artificial Intelligence Review 39* (2011), 251–260.

[9] FOGEL, D. B. Evolving strategies in blackjack. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)* (June 2004), vol. 2, pp. 1427–1434 Vol.2.

[10] KENDALL, G., AND SMITH, C. The evolution of blackjack strategies. vol. 4, pp. 2474 – 2481 Vol.4.

[11] LIANG, J., MEYERSON, E., HODJAT, B., FINK, D., MUTCH, K., AND MIIKKULAINEN, R. Evolutionary neural automl for deep learning, 2019.

[12] MILLMAN, M. H. A statistical analysis of casino blackjack. *The American Mathematical Monthly 90*, 7 (1983), 431–436.

[13] NISSEN, S. Large scale reinforcement learning using q-sarsa (λ) and cascading neural networks. *Unpublished masters thesis, Department of Computer Science, University of Copenhagen, København, Denmark* (2007).

[14] SUTTON, R. S., AND BARTO, A. G. *Reinforcement Learning: An Introduction*, second ed. The MIT Press, 2018.

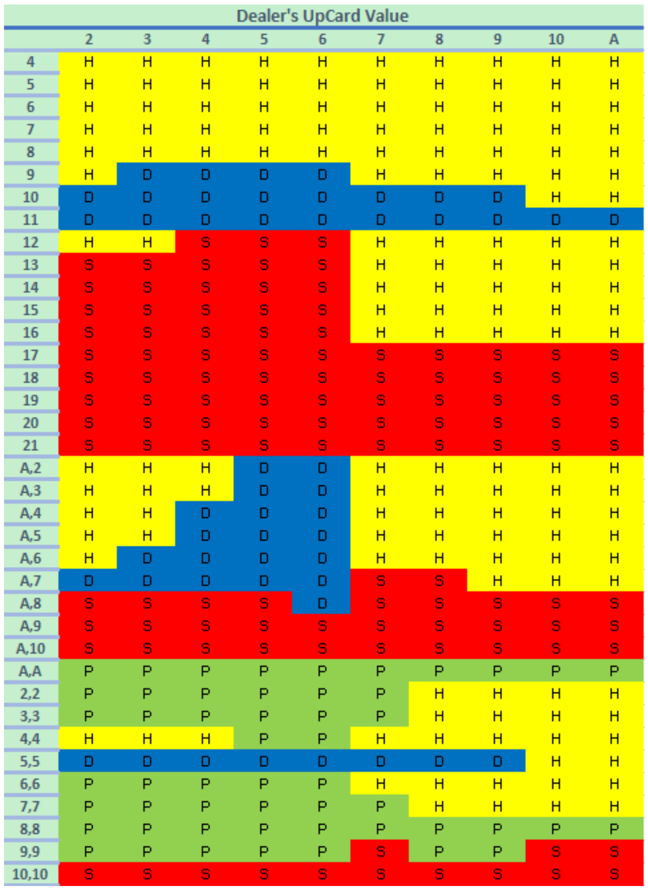
[15] THORP, E. O. *Beat the Dealer: a winning strategy for the game of twenty one*, vol. 310. Vintage, 1966.

[16] VAN DER GENUGTEN, B. B. Blackjack in holland casino's: basic, optimal and winning strategies. *Statistica Neerlandica 51*, 3 (1997), 318–344.

[17] WATKINS, C. J., AND DAYAN, P. Q-learning. *Machine learning 8*, 3-4 (1992), 279–292.

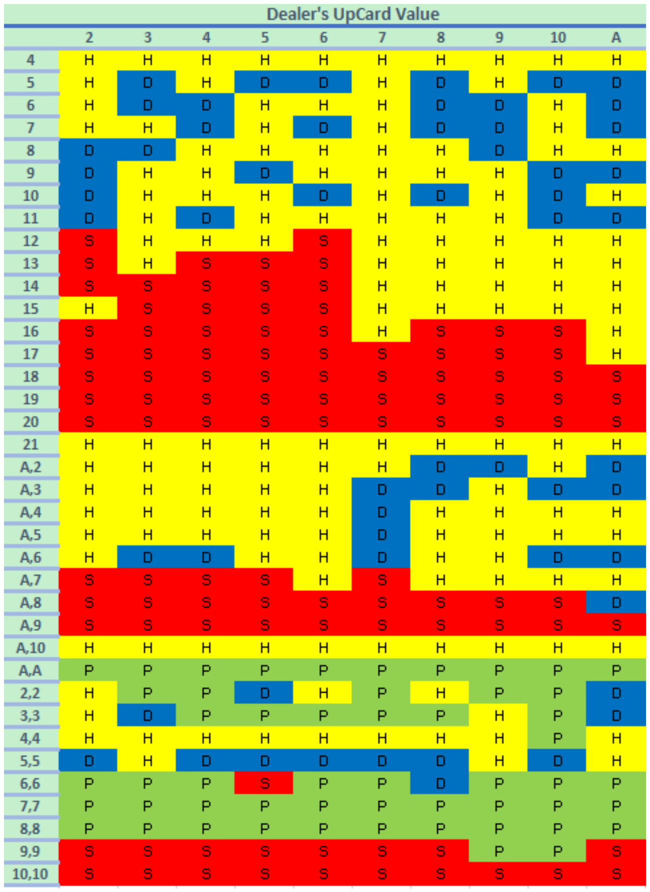
[18] WU, A. Playing blackjack with deep q-learning, 2018.

APPENDIX



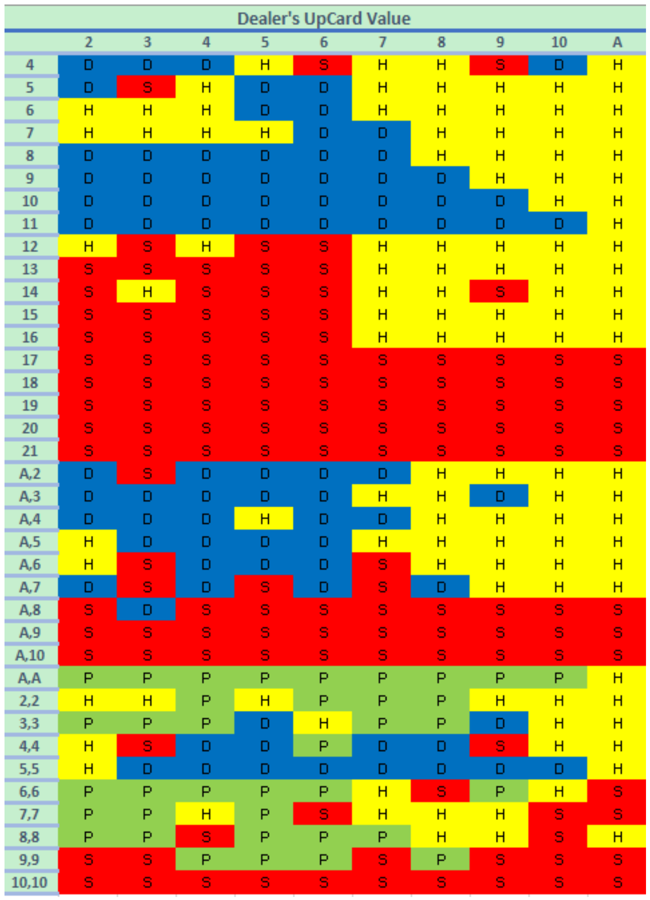
	Dealer's UpCard Value									
	2	3	4	5	6	7	8	9	10	A
4	H	H	H	H	H	H	H	H	H	H
5	H	H	H	H	H	H	H	H	H	H
6	H	H	H	H	H	H	H	H	H	H
7	H	H	H	H	H	H	H	H	H	H
8	H	H	H	H	H	H	H	H	H	H
9	H	D	D	D	D	H	H	H	H	H
10	D	D	D	D	D	D	D	D	H	H
11	D	D	D	D	D	D	D	D	D	D
12	H	H	S	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	H	H	H	H
16	S	S	S	S	S	H	H	H	H	H
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
A,2	H	H	H	D	D	H	H	H	H	H
A,3	H	H	H	D	D	H	H	H	H	H
A,4	H	H	D	D	D	H	H	H	H	H
A,5	H	H	D	D	D	H	H	H	H	H
A,6	H	D	D	D	D	H	H	H	H	H
A,7	D	D	D	D	D	S	S	H	H	H
A,8	S	S	S	S	D	S	S	S	S	S
A,9	S	S	S	S	S	S	S	S	S	S
A,10	S	S	S	S	S	S	S	S	S	S
A,A	P	P	P	P	P	P	P	P	P	P
2,2	P	P	P	P	P	P	H	H	H	H
3,3	P	P	P	P	P	P	H	H	H	H
4,4	H	H	H	P	P	H	H	H	H	H
5,5	D	D	D	D	D	D	D	D	H	H
6,6	P	P	P	P	P	H	H	H	H	H
7,7	P	P	P	P	P	P	H	H	H	H
8,8	P	P	P	P	P	P	P	P	P	P
9,9	P	P	P	P	P	S	P	P	S	S
10,10	S	S	S	S	S	S	S	S	S	S

Figure 7: Basic Strategy



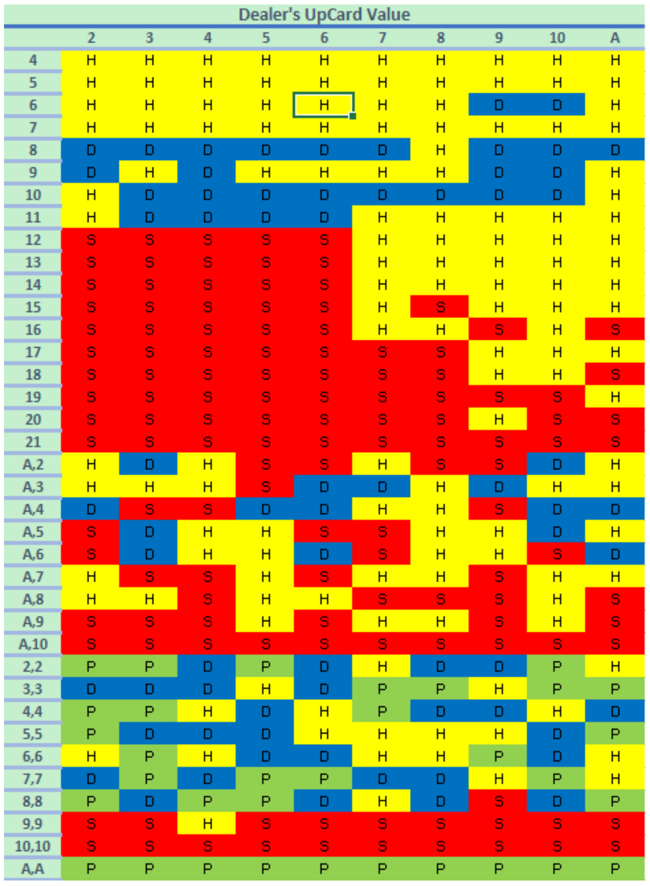
	Dealer's UpCard Value									
	2	3	4	5	6	7	8	9	10	A
4	H	H	H	H	H	H	H	H	H	H
5	H	D	H	D	D	H	D	H	D	D
6	H	D	D	H	H	H	D	D	H	D
7	H	H	D	H	D	H	D	D	H	D
8	D	D	H	H	H	H	H	D	H	H
9	D	H	H	D	H	H	H	H	D	D
10	D	H	H	H	D	H	D	H	D	H
11	D	H	D	H	H	H	H	H	D	D
12	S	H	H	H	S	H	H	H	H	H
13	S	H	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
15	H	S	S	S	S	H	H	H	H	H
16	S	S	S	S	S	H	S	S	S	H
17	S	S	S	S	S	S	S	S	S	H
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	H	H	H	H	H	H	H	H	H	H
A,2	H	H	H	H	H	H	D	D	H	D
A,3	H	H	H	H	H	D	D	H	D	D
A,4	H	H	H	H	H	D	H	H	H	H
A,5	H	H	H	H	H	D	H	H	H	H
A,6	H	D	D	H	H	D	H	H	D	D
A,7	S	S	S	S	H	S	H	H	H	H
A,8	S	S	S	S	S	S	S	S	S	D
A,9	S	S	S	S	S	S	S	S	S	S
A,10	H	H	H	H	H	H	H	H	H	H
A,A	P	P	P	P	P	P	P	P	P	P
2,2	H	P	P	D	H	P	H	P	P	D
3,3	H	D	P	P	P	P	P	H	P	D
4,4	H	H	H	H	H	H	H	H	P	H
5,5	D	H	D	D	D	D	D	H	D	H
6,6	P	P	P	S	P	P	D	P	P	P
7,7	P	P	P	P	P	P	P	P	P	P
8,8	P	P	P	P	P	P	P	P	P	P
9,9	S	S	S	S	S	S	S	P	P	S
10,10	S	S	S	S	S	S	S	S	S	S

Figure 8: Q-Learning Strategy



	Dealer's UpCard Value									
	2	3	4	5	6	7	8	9	10	A
4	D	D	D	H	S	H	H	S	D	H
5	D	S	H	D	D	H	H	H	H	H
6	H	H	H	D	D	H	H	H	H	H
7	H	H	H	H	D	D	H	H	H	H
8	D	D	D	D	D	D	H	H	H	H
9	D	D	D	D	D	D	D	H	H	H
10	D	D	D	D	D	D	D	D	H	H
11	D	D	D	D	D	D	D	D	D	H
12	H	S	H	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
14	S	H	S	S	S	H	H	S	H	H
15	S	S	S	S	S	H	H	H	H	H
16	S	S	S	S	S	H	H	H	H	H
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
A,2	D	S	D	D	D	D	H	H	H	H
A,3	D	D	D	D	D	H	H	D	H	H
A,4	D	D	D	H	D	D	H	H	H	H
A,5	H	D	D	D	D	H	H	H	H	H
A,6	H	S	D	D	D	S	H	H	H	H
A,7	D	S	D	S	D	S	D	H	H	H
A,8	S	D	S	S	S	S	S	S	S	S
A,9	S	S	S	S	S	S	S	S	S	S
A,10	S	S	S	S	S	S	S	S	S	S
A,A	P	P	P	P	P	P	P	P	P	H
2,2	H	H	P	H	P	P	P	H	H	H
3,3	P	P	P	D	H	P	P	D	H	H
4,4	H	S	D	D	P	D	D	S	H	H
5,5	H	D	D	D	D	D	D	D	D	H
6,6	P	P	P	P	P	H	S	P	H	S
7,7	P	P	H	P	S	H	H	H	S	S
8,8	P	P	S	P	P	P	H	H	S	H
9,9	S	S	P	P	P	S	P	S	S	S
10,10	S	S	S	S	S	S	S	S	S	S

Figure 9: Genetic Algorithm Strategy



	Dealer's UpCard Value									
	2	3	4	5	6	7	8	9	10	A
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	D	D	H
7	H	H	H	H	H	H	H	H	H	H
8	D	D	D	D	D	D	H	D	D	D
9	D	H	D	H	H	H	H	D	D	H
10	H	D	D	D	D	D	D	D	D	H
11	H	D	D	D	D	H	H	H	H	H
12	S	S	S	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	S	H	H	H
16	S	S	S	S	S	H	H	S	H	S
17	S	S	S	S	S	S	S	H	H	H
18	S	S	S	S	S	S	S	H	H	S
19	S	S	S	S	S	S	S	S	S	H
20	S	S	S	S	S	S	S	H	S	S
21	S	S	S	S	S	S	S	S	S	S
A,2	H	D	H	S	S	H	S	S	D	H
A,3	H	H	H	S	D	D	H	D	H	H
A,4	D	S	S	D	D	H	H	S	D	D
A,5	S	D	H	H	S	S	H	H	D	H
A,6	S	D	H	H	D	S	H	H	S	D
A,7	H	S	S	H	S	H	H	S	H	H
A,8	H	H	S	H	H	S	S	S	H	S
A,9	S	S	S	H	S	H	H	S	H	S
A,10	S	S	S	S	S	S	S	S	S	S
2,2	P	P	D	P	D	H	D	D	P	H
3,3	D	D	D	H	D	P	P	H	P	P
4,4	P	P	H	D	H	P	D	D	H	D
5,5	P	D	D	D	H	H	H	H	D	P
6,6	H	P	H	D	D	H	H	P	D	H
7,7	D	P	D	P	P	D	D	H	P	H
8,8	P	D	P	P	D	H	D	S	D	P
9,9	S	S	H	S	S	S	S	S	S	S
10,10	S	S	S	S	S	S	S	S	S	S
A,A	P	P	P	P	P	P	P	P	P	P

Figure 10: ENN Strategy