Project: Texas Hold'em Poker AI

Minh Tran - tran0842

May 10, 2021

Abstract

This project paper aims to discuss the design of a simple Artificial Intelligence that is capable of playing Texas Hold'em Poker. Decision making process and certain fields are kept limited to make the implementation of the design possible. A working Python version of design is accomplished, featuring a player vs AI and an AI vs AI mode. Many quality of life improvements can be made, and the design itself is open for future expansion.

1 Introduction

Poker, along with its many variations, is one of the most popular card game in the world, both in a recreational and in a professional sports sense; it is many people's favorite past time activity, and at the same time a sport that have multi-million Dollars prize pool every year. Poker is also one of the few games that have received much attention from computer science researchers and there has been steady progress in making AIs capable of mastering the game.

Poker, in essence, is a card game where players possess a fixed number of cards on hand and they would bet on their cards to see who has the highest rank among them. Whoever has the highest rank from their cards combination when revealed then wins the "pot", the namesake for all the wagers collected from the players. As much as it is luck based, since the player would want to possess a good hand of cards, Poker is also skill-based as players often engage in a form of "psychological warfare", where players may try to bluff their way into victory, even when their hands are not that good, and other players would try to call out their bluffs.

Currently there already exist Artificial Intelligence programs that are capable of playing Poker of certain variations. For this project, I want to focus on developing a rudimentary AI that is capable of playing Texas Hold'em Poker. Why only rudimentary , one may ask, and that is because if I were to create an AI that is capable of playing poker with all its complexity, then the task would be too grand for one person to tackle, not to mention that there arise potential problems that would require knowledge that have not been taught in class yet. In some Poker programs that are mentioned here, they either have fixed betting and/or tend to limit themselves with a fixed number of players, and in most cases only two players are allowed including the machine player to further reduce complexity like some of the AIs that are mentioned below. What I will do, in addition to fixed betting, is to have the AI focus on betting solely based on its likelihood of winning that match while ignoring its opponent's wager.

2 Literature Review

An Overview of Poker's Probabilities

Texas Hold'em Poker is a game of imperfect information, in that the agent has no way of knowing perfectly what hands the opponents are holding, nor the cards that are dealt as the community cards. For most professional players, relying on luck is not an optimal strategy as they tend to calculate the probabilities of what hands their opponents have combined with evaluating the potential value of their hand before making a decision. Poker probability is stochastic in nature, thus different hands, different community cards all contribute to a wide range of possible winning possibilities [2]. It is not without mention that the agent may also have to account the bet made by their opponent in addition to gauging their hand's strength.

AIs in Poker

a. SARTRE

SARTRE [6] is an AI developed for 2-player Texas hold'em using a memory based system, in which its machine learning algorithm collect and review over poker matches of strong players and then make plays based on the experience learned based on such information. The AI when in play evaluates its hand's strength in combination with the community card by categorizing both the current hand's strength and its potential to improve in subsequent rounds. Also, the AI evaluates and represents the betting decision in the form of a a decision tree corresponding to the three possible decisions in each betting round. The memory system, on the other hand, having the knowledge of professional matches, acts like a consulting partner to the AI as it check which decisions is the best one make.

b. DeepStack

DeepStack [5] is one of the more recent AI to emerge, also competing within the same poker category as SARTRE, but with the twist that DeepStack has to play with no limit bets (essentially no cap on how large the bet can be), where the above AI has such a restriction. DeepStack team boasts of coming up with a capable heuristic search that is applicable to an imperfect information system, which is the heart and soul of its poker algorithm. The algorithm goal is to come up with an approximate Nash equilibrium to base the AI's decision on. At the same time, the AI relies on its deep neural network (deep learning is a concept that we have not really discussed in the class yet) that, from my understanding, generates values used for evaluation of potential future state of the game, and the process gets better through many self-play which then becomes the AI's "intuition".

c. Pluribus

The biggest difference between Pluribus [3] and the above AIs is that Pluribus is built for six-player no-limit Texas hold'em Poker, not for two-player. As such, Pluribus must devise new strategies to deal with multiple opponents to account for the changing probability caused by an increase in player count. The AI's core idea is about improving itself through self-play using deep learning similar to DeepStack. The strategy Pluribus gets from self-play is called the blueprint strategy which is supposed to help with reducing process time, and when the game may deviate from the blueprint or reach to a certain point, Pluribus then uses real-time search to find a more optimal strategy in response. The AI, depending on the circumstances, may switch between different version of counterfactual regret minimization techniques to determine the suitable strategy for each subsequent sub-games within the depth-limited search.

Game AIs Outside of Poker

Sometimes, taking a look at sources outside of the current specific field, which is Poker for this project, may prove useful in finding what is working in Artificial Intelligence in general. Within the recent years, researchers have come up with new AIs that are able to beat profession players in Go [7], a much more complicated game move-wise compared to chess, and in StarCraft 2 [1] and DOTA 2 [8], where both games have imperfect information compared to the two previous mentioned games.

The mentioned AIs all make used of deep neural network learning, in which the core idea is similar to the Poker AIs DeepStack and Pluribus. Thus it can be seen how powerful deep learning can make when it is applied to any AI, and that the lack of it in a program makes it much less robust when it comes to decisions making, both response time and logic wise.

3 Approaching the Problem

A standard game of Texas Hold'em Poker can be divided into two distinct problems, being the evaluation of the player's current hole cards (the two cards that are dealt to the player) given the current state of the board, and the decision of what to do next (raise, check, call, and fold). The former problem is easier to solve while the latter is less so, but both problems are still complicated regardless, and thus require an in-depth examination.

Evaluating a player's hole cards means two questions: (1) what is the best combination of cards possible that can be constructed from using the hold cards and cards on the board, and (2) what is the likelihood of the player's hand strength being higher than that of their opponent? My first and current approach to the first question is to use a bit-wise evaluator where the value of a card is represented as a string of bits of the size four bytes, and the evaluation of hand strength is a series of bit-wise operations and logic statements. The codes I used for this part is sourced from a variation of Will Drevo's code [4]. My second approach was to construct my own evaluator where it has card as a class of two main variables, one being the value of the card, and the other being the suit of the card. This approach, in theory, would only require logic statements for evaluation, but then due to a constraint of time, it is left unfinished to finish the Poker AI. My approach to the second problem is to simulate all the possible combinations of hole cards that the opponent can have, then compare each combination's hand strength to the player's, then record the results. The final output is the a ratio of the number of wins over the total number of comparisons. Here is the pseudo-code for my algorithm:

Algorithm 1: Calculate the odds of winning given the hole cards and the board

```
function OddsCalculator ($hand, board, deck$);
wins , losses , draws = 0;
handStrength = evaluate($hand, board$);
for $holeCards \in$ combination($2, board$) do
    possibleOpponentHS = evaluate($holeCards, board$);
    // The lower the value of handStrength, the more likely it
       is to win
    if handStrength < possibleOpponentHS then
        wins = wins +1;
    else if handStrength > possibleOpponentHS then
        losses = losses +1;
    else
        draws = draws +1;
end
odds = wins / (wins + losses + draws );
return odds;
```

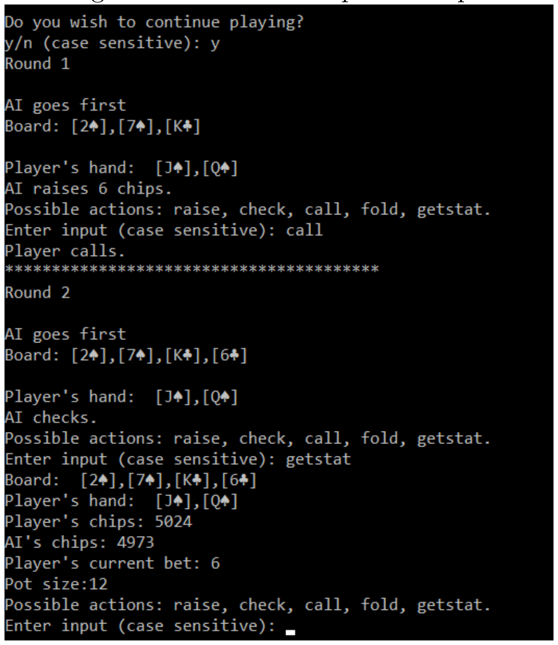
With regards to the betting aspect of the game, a player bets based on two main pieces of information: the player's hand strength, and the action made by the opponent. Using the win percentage that can be calculated with the above algorithm, the AI can do a set of different actions corresponding to a wide range of thresholds. For example, an AI having a winning odd of 0.875 is more likely to make riskier wager, while an AI having a winning odd of 0.462 is more likely to go for either small bets or just call or check. This can be done using lots of if-else statements and coming up with a sound enough strategy for the AI to use. On the other hand, the psychological aspect of the decision making is much more difficult to implement. In a game of Poker, a high wager means that either the player's hand is strong, or that they are trying to bluff, To accommodate this, most development teams opt for using machine learning to teach the AI by feeding data of thousands and more Poker matches so that it can learn the optimal behaviors when making decisions. I couldn't come up with any simple solution to this particular problem and thus decided to strip it completely, hence the simplification of my Poker AI.

4 Experiment Design and Results

Since the aim of this project is to build a functional rudimentary AI that is capable of playing Texas Hold'em Poker, certain fields and actions are purposefully kept limited. For one, I coded it so that both the human player and the AI have a large amount of chips, and the bet size is kept limited per wager (which also means no All-In) to ensure that it will be extremely difficult for either player to run out of chips, unless a lot of matches are continuously played.

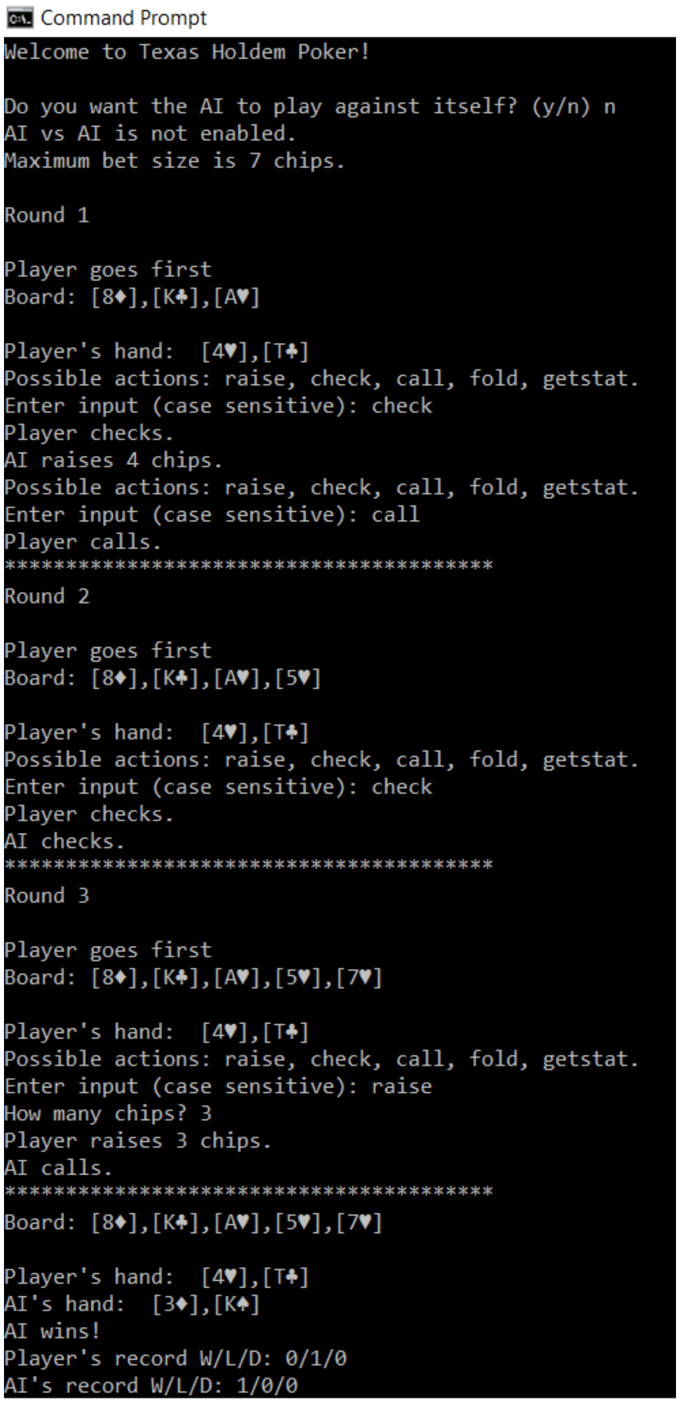
A large portion of the code for this project had to be written to accommodate the player's input and the user interface, which includes error checking for rule-breaking action (a check cannot be followed by a call), or when an extremely large-bet is made.

Figure 1: Terminal Output Example



```
Do you wish to continue playing?
y/n (case sensitive): y
Round 1

AI goes first
Board: [2♠],[7♠],[K♣]

Player's hand:  [J♠],[Q♠]
AI raises 6 chips.
Possible actions: raise, check, call, fold, getstat.
Enter input (case sensitive): call
Player calls.
****************************************
Round 2

AI goes first
Board: [2♠],[7♠],[K♣],[6♣]

Player's hand:  [J♠],[Q♠]
AI checks.
Possible actions: raise, check, call, fold, getstat.
Enter input (case sensitive): getstat
Board:  [2♠],[7♠],[K♣],[6♣]
Player's hand:  [J♠],[Q♠]
Player's chips: 5024
AI's chips: 4973
Player's current bet: 6
Pot size:12
Possible actions: raise, check, call, fold, getstat.
Enter input (case sensitive): 
```

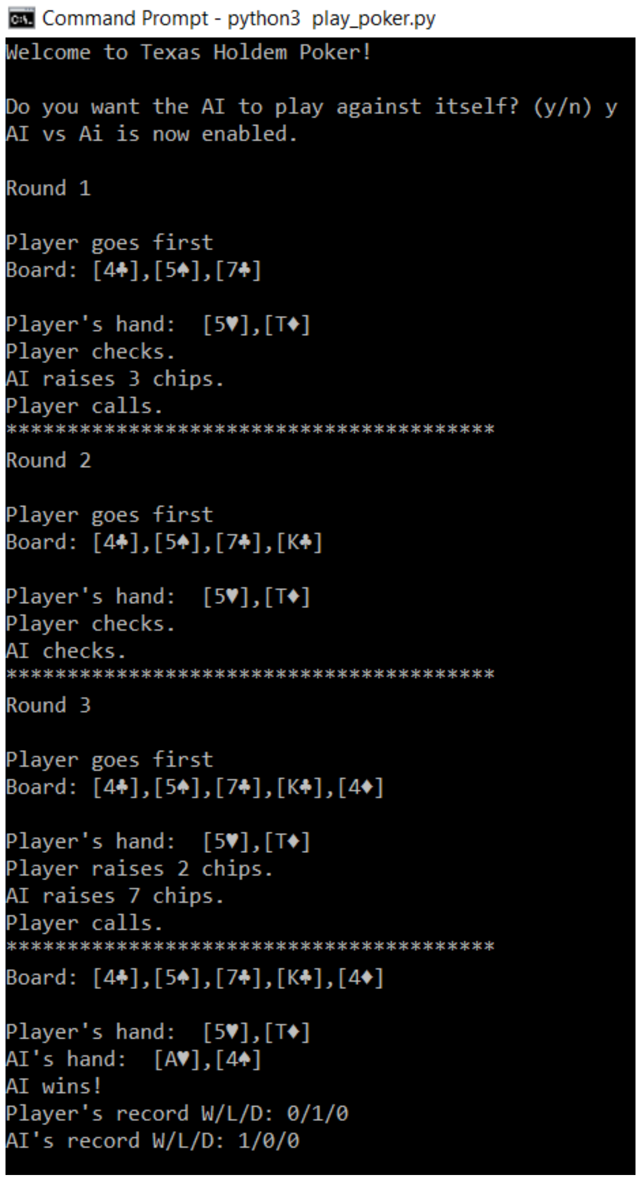
Figure 2: Full Match Player vs AI Example



Command Prompt

```
Welcome to Texas Holdem Poker!

Do you want the AI to play against itself? (y/n) n
AI vs AI is not enabled.
Maximum bet size is 7 chips.

Round 1

Player goes first
Board: [8♦],[K♣],[A♥]

Player's hand:  [4♥],[T♣]
Possible actions: raise, check, call, fold, getstat.
Enter input (case sensitive): check
Player checks.
AI raises 4 chips.
Possible actions: raise, check, call, fold, getstat.
Enter input (case sensitive): call
Player calls.
****************************************
Round 2

Player goes first
Board: [8♦],[K♣],[A♥],[5♥]

Player's hand:  [4♥],[T♣]
Possible actions: raise, check, call, fold, getstat.
Enter input (case sensitive): check
Player checks.
AI checks.
****************************************
Round 3

Player goes first
Board: [8♦],[K♣],[A♥],[5♥],[7♥]

Player's hand:  [4♥],[T♣]
Possible actions: raise, check, call, fold, getstat.
Enter input (case sensitive): raise
How many chips? 3
Player raises 3 chips.
AI calls.
****************************************
Board: [8♦],[K♣],[A♥],[5♥],[7♥]

Player's hand:  [4♥],[T♣]
AI's hand:  [3♦],[K♠]
AI wins!
Player's record W/L/D: 0/1/0
AI's record W/L/D: 1/0/0
```

Figure 3: Full Match AI vs AI Example



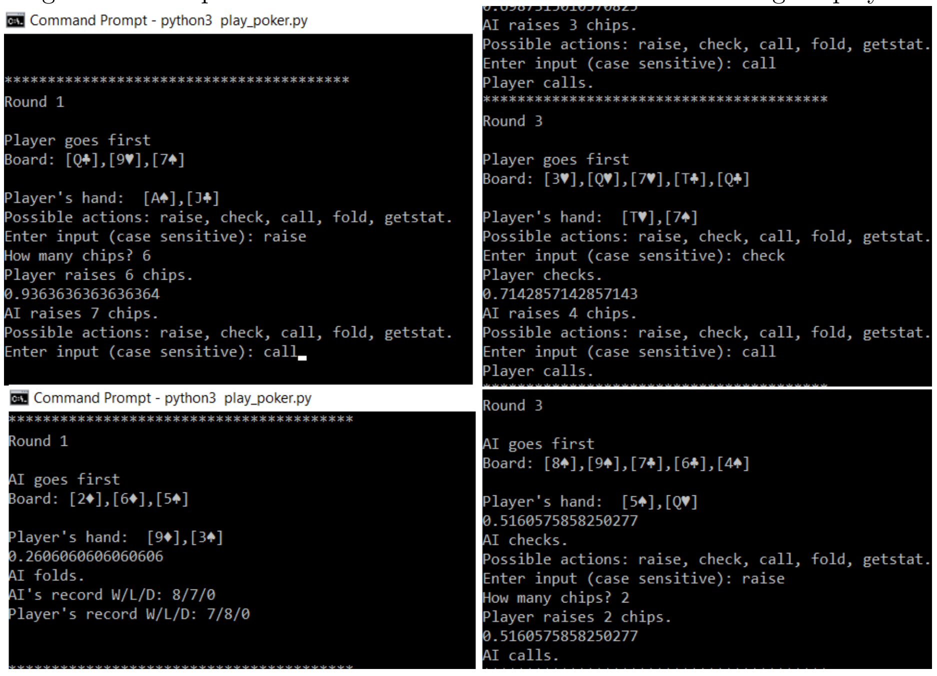
```
Command Prompt - python3  play_poker.py
Welcome to Texas Holdem Poker!

Do you want the AI to play against itself? (y/n) y
AI vs Ai is now enabled.

Round 1

Player goes first
Board: [4♣],[5♠],[7♣]

Player's hand:  [5♥],[T♦]
Player checks.
AI raises 3 chips.
Player calls.
****************************************
Round 2

Player goes first
Board: [4♣],[5♠],[7♣],[K♣]

Player's hand:  [5♥],[T♦]
Player checks.
AI checks.
****************************************
Round 3

Player goes first
Board: [4♣],[5♠],[7♣],[K♣],[4♦]

Player's hand:  [5♥],[T♦]
Player raises 2 chips.
AI raises 7 chips.
Player calls.
****************************************
Board: [4♣],[5♠],[7♣],[K♣],[4♦]

Player's hand:  [5♥],[T♦]
AI's hand:  [A♥],[4♠]
AI wins!
Player's record W/L/D: 0/1/0
AI's record W/L/D: 1/0/0
```

5 Result Analysis

It is hard to do a complete measure when it concerns not only some algorithms, but also the whole program itself. The main objective of this project is to build a simple AI that is capable of playing Poker, and by running the program numerous times, it seems that the objective has been met without any major problems. Of course, further testings are needed to examine the AI as there was only a limited number of testers; I was one, and I asked a friend of mine who knows how to play Poker to test it out, thus making it two testers. Therefore there might be potential issues that have not been discovered yet, but as of the time of this writing, the program runs quite smoothly. It is very recommended that whoever is reading this paper should try out playing with the AI with the code going along with this paper.

After multiple matches with AI, some against human players and some against itself, the strategy that is hard-coded for the AI to follow seems to be working without issues. In figure 4, it can be observed that when the winning odd is high, the AI is much more aggressive, being shown through high bet count, but when the odds are low, the AI is much more conservative with its actions. opting more for checks and calls, and if the odd is too low, it will simply fold most of the time.

Figure 4: Examples of Actions AI Took with Its Odds of Winning Displayed



Command Prompt - python3 play_poker.py

```
****************************************
Round 1

Player goes first
Board: [Q♣],[9♥],[7♠]

Player's hand:  [A♠],[J♣]
Possible actions: raise, check, call, fold, getstat.
Enter input (case sensitive): raise
How many chips? 6
Player raises 6 chips.
0.9363636363636364
AI raises 7 chips.
Possible actions: raise, check, call, fold, getstat.
Enter input (case sensitive): call_
```

```
0.6987315010570825
AI raises 3 chips.
Possible actions: raise, check, call, fold, getstat.
Enter input (case sensitive): call
Player calls.
****************************************
Round 3

Player goes first
Board: [3♥],[Q♥],[7♥],[T♣],[Q♣]

Player's hand:  [T♥],[7♠]
Possible actions: raise, check, call, fold, getstat.
Enter input (case sensitive): check
Player checks.
0.7142857142857143
AI raises 4 chips.
Possible actions: raise, check, call, fold, getstat.
Enter input (case sensitive): call
Player calls.
****************************************
```

Command Prompt - python3 play_poker.py

```
****************************************
Round 1

AI goes first
Board: [2♦],[6♦],[5♠]

Player's hand:  [9♦],[3♠]
0.2606060606060606
AI folds.
AI's record W/L/D: 8/7/0
Player's record W/L/D: 7/8/0


****************************************
```

```
Round 3

AI goes first
Board: [8♠],[9♠],[7♣],[6♣],[4♠]

Player's hand:  [5♠],[Q♥]
0.5160575858250277
AI checks.
Possible actions: raise, check, call, fold, getstat.
Enter input (case sensitive): raise
How many chips? 2
Player raises 2 chips.
0.5160575858250277
AI calls.
****************************************
```

Since the the cards and consequently the card combinations are represented as strings of bits, the speed of the hand strength comparisons is almost instantaneous as algorithm 1 has a time complexity of $O(n)$ and there is not much data to simulate through. Coupled with the simplified AI's strategy in making a decision, the computation speed is thus understandable. On the other hand it would be better to record the time it takes for the AI to decide an action and contrast the recorded data to publicly available Poker AI. But the aim of the project is not to test the efficacy of the created AI but simply to construct a working AI. The measuring of the AI's efficiency can be one of the next checkpoint when the AI is further refined into something more complicated.

Two problems that arise from this design are that the AI's reaction time does not reflect a realistic depiction of a human player, and that the strategy is only both simple and singular. The first problem might seem obvious when stated: after all, the computer is superior to humans when it comes to sheer calculating power. But one often overlooked aspect when it comes to this type of games is that the time player spent thinking is also the time their opponent spent thinking. A simple fix is to have a random time variable for the AI to make an action, which makes it more realistic superficially, or we can have the AI actually simulate enough scenarios ahead of time so that the computational time may take as long as the thinking time of a human player. The second option is infinitely more complicated, but is much more rewarding since it would improve the playing capacity of the AI. The second problem is the lack of variety in Poker strategy for the AI. Having one simple and singular strategy makes the AI's play style becomes predictable, and thus given enough time any human player can notice a pattern and try to play around it. A potential extension without having to resort to implementing an ever-dynamic strategy is to construct several simple strategies that can be interchanged, either randomly or based on number of matches played or bets made. While tedious to code, this option is much more doable without having to rely on deep learning like other Poker AIs that are mentioned in the literature review section.

6 Conclusion

This project paper gives a general design of an AI that aims to play Texas Hold'em Poker with in-depth discussions on how the AI would approach playing the game, including strategy it uses to play the game and the algorithm that handles evaluating the its winning odds. Then the project provides examples of how the AI plays against human players and even against itself. All in all, the implementation of the design discussed here is a success, but it is still far from being a good Poker AI as it faces some limitations to make the implementation becomes possible. The next step for this project would be either to try a non-bit representation of cards and compare it to the current approach, or to increase the robustness of the AI's strategic capacity.

References

[1] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. Pinto, J. Raiman, T. Salimans, J. Schlatter, and S. Zhang. Dota 2 with large scale deep reinforcement learning. 12 2019.

[2] D. Bragonier. Statistical analysis of texas holdem poker. *Statistics*, 06 2010.

[3] N. Brown and T. Sandholm. Superhuman ai for multiplayer poker. *Science*, 365:eaay2400, 07 2019.

[4] I. Hendley. Treys. https://github.com/ihendley/treys, 2019.

[5] M. Moravcik, M. Schmid, N. Burch, V. Lisý, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, and M. Bowling. Deepstack: Expert-level artificial intelligence in no-limit poker. *Science*, 356, 01 2017.

[6] J. Rubin and I. Watson. A memory-based approach to two-player texas hold'em. In *AI '09: Proceedings of the 22nd Australasian Joint Conference on Advances in Artificial Intelligence*, pages 465–474, 12 2009.

[7] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 01 2016.

[8] O. Vinyals, I. Babuschkin, W. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. Agapiou, M. Jaderberg, and D. Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575, 11 2019.