AlphaBluff: An AI-Powered Heads-Up No-Limit Texas Hold'em Poker Video Game

Aristotelis Lazaridis, Christos Perchanidis, Doxakis Chovardas and Ioannis Vlahavas
School of Informatics
Aristotle University of Thessaloniki
Email: arislaza@csd.auth.gr

***Abstract*—Complex games require disparate behaviors in order to be solved, giving space to researchers to study AI model behaviors in various settings. At the same time, the video game industry benefits by incorporating these models in their games for delivering realistic and challenging gameplay experience to users. However, there is a well-known difficulty of implementing and training efficient models in complex games for entertainment purposes. In this paper, we report on our approach to overcome this challenge and ultimately develop AlphaBluff, a Heads-Up No-Limit Texas Hold'em (HUNL) Poker variation video game developed in the Unity game engine, in which human players can play against trained AI opponents. Initially we trained different state-of-the-art AI models and analyzed their individual performance scores in a custom HUNL environment, as well as their performance against each other. AlphaBluff was developed with the goal of producing a professional-level poker video game that includes cutting-edge AI opponents, which, to our knowledge has never been developed before. Using data gathered by gameplay sessions from beta testers, we performed a statistical analysis and concluded that our models have a high win rate against human players. An adaptation of our system can further enrich this unique gameplay experience by combining these models, data and statistical reports, in order to develop an efficient player-opponent matchmaking mechanism.**

***Index Terms*—artificial intelligence, reinforcement learning, video games, poker**

I. Introduction

The variety of game types and their individual mechanics pose different challenges for Artificial Intelligence (AI) when it comes to solving them. Several methods have been developed that achieve high gameplay performance in various games, with the most prominent example being techniques that belong to the field Deep Reinforcement Learning [1].

AI techniques are continuously applied to complex games in order to solve them, which is the natural path of research in this field since solving simpler games meant that the bar had to be raised higher to extend further the boundaries of AI. To this end, Poker and its variations were always targets of high value for researchers due to their immense complexity produced by their imperfect-information nature [2]. Thus, not many successful attempts at producing a high-performance model in this popular game (and its variations) exist to date. Admittedly, even the most eminent cases of such AI systems require enormous amounts of processing power in order to train adequately and become top-tier players. As such, this also makes it extremely difficult to even train and compare existing models, much less use them as computer-controlled opponents in custom video games for the purposes of providing a modern, challenging and realistic user gameplay experience.

In this paper, we present an approach for tackling these issues with the end goal of producing a professional Heads-Up No-Limit (HUNL) Texas Hold'em poker video game that incorporates state-of-the-art AI algorithms as opponents to human players. We trained and evaluated 7 different Poker AI agents that have different learning mechanisms. Since training these algorithms require enormous resources to reach the level of an experienced poker player, we used our available computing power as optimally as we could in order to produce models that could perform decently on average.

Additionally, for this paper we also developed AlphaBluff[1] (Figure 1), a modern 3D Texas Hold'em Poker video game created in the Unity game engine, in which human players can play against four types of AI opponents. Apart from developing and releasing AlphaBluff for entertainment purposes, it also serves as a means to evaluate our trained models against human players, with statistical results indicating that they can reach a win rate of about 73% against human players.

II. Related work

Poker has been in the center of attention right after AI set its footprint in the area of games, but its high level of complexity and hidden-information nature proved to be an obstacle in delivering a system that is capable of winning against experts. Since early on, the most straightforward method of developing a Poker AI agent was by using rule-based systems built by domain experts [3].

However, the most promising attempts arrived with the use of Search techniques. More specifically, the first AI agent that achieved victories against professional rank poker players in HUNL is DeepStack [4], which incorporates a technique known as Counter-Factual Regret (CFR) minimization algorithm [5]. Libratus [6], a powerful Poker-playing system, managed to win against top-tier rank players in HUNL, and its successor, Pluribus [7], was developed and trained in multiplayer No-Limit Texas Hold'em. Recursive Belief-Based Learning (ReBeL) [8] is the most recent Poker AI that achieves superhuman performance in HUNL, and it can be used for perfect and imperfect-information games. Due to the resource-heavy processes in all of these models, the focus in other works

[1] https://www.youtube.com/watch?v=oFacNlXe1LQ

Fig. 1. AlphaBluff in-game screenshot.

(e.g. AlphaHoldem [9]) is on reducing the computational power required in various ways while optimizing performance, or improving the generalization abilities of models (e.g. [10]).

To the best of our knowledge, there have been no attempts in conducting a cross-comparison evaluation of different algorithms in the game of Poker, which is partially one of the main contributions of this paper, in order to provide baseline performance indications within the same environment.

III. Background

Poker is a hidden-information card game with 52 cards on the standard deck. Each card belongs in one of 4 suits (clubs, diamonds, hearts and spades) and has one of 13 ranks (2 – 10, Jack, Queen, King, Ace). Players wager who has the better hand in each game, according to the game rules. Each game's winner of each game receives all bets that have been wagered. While several variations of Poker exist, Texas Hold'em is the most well-known, which is the specific variation that we used in our work. Among the Poker variations, No-Limit Texas Hold'em is one of the most difficult for AI systems to solve, in which a player can bet any amount of chips, including all remaining stake (all-in). Heads-Up No-Limit (HUNL) Texas Hold'em refers to the variation in which only two players participate.

A. AI Methods in Poker

The most common approach, and the one with the best results, is trying to approximate the Nash Equilibrium using principles for the scientific field of Game Theory. One method of computing Nash Equilibrium is through a method called Fictitious Play [11]. Fictitious Self-Play (FSP) [12] is a method based on the same principles with Fictitious Play but also takes into account the concept of time and sequence in games. A proposed improvement of FSP is Neural Fictitious Self-Play (NFSP) [13], which combines FSP and Neural Networks in order to enhance its performance. It consists of two independent Neural Networks and a memory buffer in each of them; the first network tries to predict action values from stored data in its memory buffer, and the second network maps the game's states to action probabilities using supervised learning, subsequently defining the agent's average strategy. NFSP is scalable, it can work without prior domain knowledge, and it was proven that it approaches Nash Equilibrium in small poker games while being competitive in Limit Texas Hold'em, a much larger game.

Another important method for approximating Nash Equilibrium in imperfect information games is Counterfactual Regret Minimization (CFR) [5], and a variation termed Deep CFR is the first non-tabular CFR algorithm that achieves strong performance in massive poker games like HUNL [14].

IV. Materials and Methods

Our implementation consists of the RL-formulated HUNL environment, the algorithms that were used for the training and evaluation procedure, the Unity-based AlphaBluff GUI that was developed, and a brief statistical report on the performance of our agents against human players.

A. HUNL Environment

A custom HUNL environment was implemented and formulated as a Reinforcement Learning problem, inspired by the environment implemented in RLCard [15]. In that environment, each player has 100 chips as his initial stack, the big blind's size is 2 chips, while the small blind is 1 chip. After the end of each hand, the game is reset and the players start again with their initial stack size. This variation is called "Doyle's Game", and it is commonly used by the scientific community.

This HUNL environment is abstracted from the perspective of the action space since it allows only three betting sizes: *half*

pot, *whole pot* and *all-in*. Along with the *fold*, *check* and *call* actions, the length of the environment's action vector is 6.

Our proposed observation space representation, which was inspired by [13], is comprised of three parts. The first part represents the betting history and is a 18x6 tensor. In this environment the maximum length of the action sequence is 18, while there are 6 actions in total, but not all of them are legal actions in each phase of the game. The second and third parts represent the hole and public cards, respectively, i.e. a k-by-n encoding in a 52-length array for each phase. The second part that encodes each player's hole cards is different for each player as it is the only part representing hidden information. The three arrays for each component are flattened and concatenated into a 316-length array. This state representation is more complex than the original, but it provides more information for the agents.

B. Implemented Agents

We implemented and evaluated different AI methods so as to understand their potential and limitations in games with imperfect information.

1) Hand Evaluator with Handcrafted Strategy: A library that calculates the winning odds based on the player's hand is used, which simulates all possible hands or approximates this probability by performing Monte Carlo simulations. For the preflop phase, it performs 1000 simulations that produce a probability very close to the actual one, while for all phases that follow, it computes the exact winning probability.

2) Proximal Policy Optimization: The PPO algorithm was chosen from the category of Deep RL algorithms. RLlib [16] was used to implement the algorithm, which is an open-source RL library. RLlib works on top of Ray [17], a python API for building distributed applications.

Since HUNL is a two-player game, two different PPO agents played against each other in the training phase. The training was performed in a self-play manner: at the end of each iteration, the agent's weights that had won (average reward) were transferred to the agent that lost. Finally, a parametric model was used in order to allow the agent to play only the legal actions in each turn.

3) Information Set Monte Carlo Tree Search (Infoset MCTS): Information set (Infoset) MCTS [18] is a variation of the MCTS implementation used for perfect information games. Infoset MCTS is designed for imperfect information games, where the player does not know in which state he is at a given moment due to hidden information. The main difference with the original algorithm is that Infosets are used instead of states. At the start of each simulation, the agent randomly chooses a state that belongs to the known Infoset and moves down the game tree, takes the payoff for that state at the leaf node and backpropagates its value until it reaches the root node. The algorithm is conducting an online search, so it does not need to be trained. For each decision, Infoset MCTS performed 1000 iterations.

4) AlphaZero with Infoset MCTS: AlphaZero [19] is a general RL algorithm that combines RL and Search. It is the successor of AlphaGo and AlphaGoZero, and it can play Chess, Go, and Shogi at a superhuman level. It uses the MCTS algorithm in order to traverse the game tree, and it utilizes two separate Deep Neural Networks. The first is used to predict the value of a state, while the other predicts the action policy. AlphaZero can handle only perfect information games, it was decided to change the MCTS with Infoset MCTS in order to test whether it can handle uncertainty better and to what extend. After each network training, an evaluation with the previous best model was performed in 100 games, and if the new one performs better then it is saved, otherwise it is discarded.

5) Neural Fictitious Self-Play (NFSP): The NFSP implementation used for the agent was provided by OpenSpiel [20], a framework built to assist researchers in the field of Reinforcement Learning in games.

6) Deep CFR: The implementation of Deep CFR was also provided by OpenSpiel. The agent performed 10 iterations of 8K traversals for each of the two players.

It should be noted that computational and other technical limitations on either server were present, allowing us to train the agents for no more than 48 hours consecutively, which constrained our experimental procedures significantly.

C. Evaluation

Approximating Nash Equilibrium is currently the best approach for creating competitive poker agents, therefore the best way to evaluate these agents is by measuring their exploitability [21], which measures how far the agent's strategy is from Nash Equilibrium.

Computing the exploitability of an agent is practically infeasible for large games like HUNL, therefore we decided to measure the exploitability of an agent with the Local Best Response (LBR) [22] method. The main concept of LBR is to compute the probability for each of the possible card combinations that the player holds as his hole cards.

To make reliable assumptions, a considerable number of games must be played by the tested agent, so as to then compute the mean reward of the LBR agent. While this method is an approximation it is still highly demanding (e.g. computing an agent's exploitability against LBR in 1000 games requires roughly 24 hours). Thus, it was decided to evaluate the agents against an opponent that chooses its next actions randomly during the training phase to monitor the training progress.

After training, all agents played against each other for 1000 evaluation games, and their mean reward was computed. Also, the agents played against a random player and a "fish" player, who always checks or calls. The LBR method was only applied against PPO, NFSP and Deep CFR agents, since these agents decide their next action simultaneously, while the other agents use search-based techniques to operate and thus their evaluation with LBR was infeasible for computational reasons.

Moreover, three other static agents (random, "fish" and "always fold" agents) were tested against LBR so as to compare their exploitability with the other agents. In the

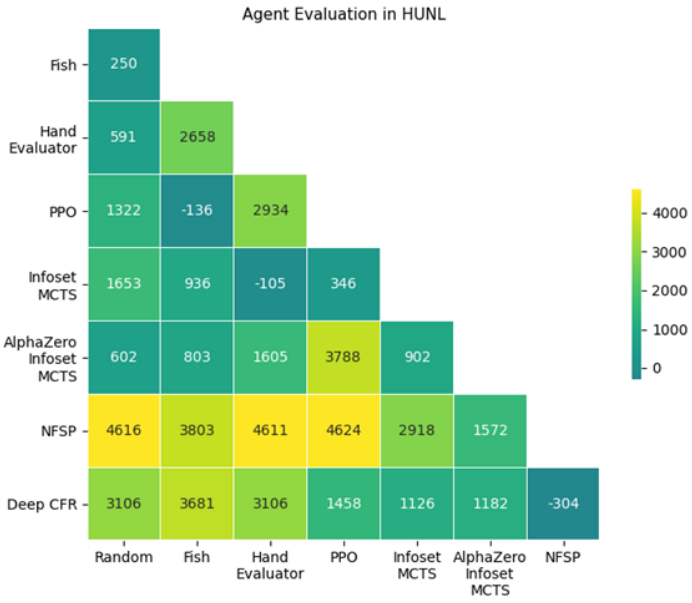

Fig. 2. Heatmap of performance scores in 1-vs-1 games between implemented agents, measured in mbb/h.

evaluation phase, their performance was measured in milli-big-blinds per hand (mbb/h), which is the one-thousandth of a big blind that players must bet at the beginning of each game and it is a standard measure of win rate in poker.

V. AI Training Results

In this section, the results of the evaluation procedure are presented (Table I and Figure 2) and discussed.

Hand Evaluator with Handcrafted Strategy. The Hand evaluator agent had the worst performance of all the agents agents. It had a negative mean reward against all but Infoset MCTS with a very close margin which is not statistically important. The hand evaluator had a positive mean reward against random and "fish" opponents, with the second being especially high. This can be explained by the fact that the Hand evaluator calculates the winning chances and so when they are high it places more bets, and the fish player only calls. A more careful creation of the handcrafted strategy by a domain expert would have boosted the agent's performance.

Fig. 3. Learning curve of the PPO algorithm during training.

Proximal Policy Optimization (PPO). PPO agent had a better mean reward against the random agent than the Hand Evaluator, but it had a margin negative mean reward against Fish. These results show that general RL methods, such as PPO, cannot handle imperfect information games properly. The learning curve of the PPO agent is shown in Figure 3.

Infoset MCTS. Infoset MCTS agent achieved positive rewards with random and fish agents and it closely won against PPO, but it had negative rewards against the rest of the agents (a close call against hand evaluator). Theoretically an increased number of simulations would have achieved better results but in practice, there have been no sign of improvement that would worth the increase in execution time.

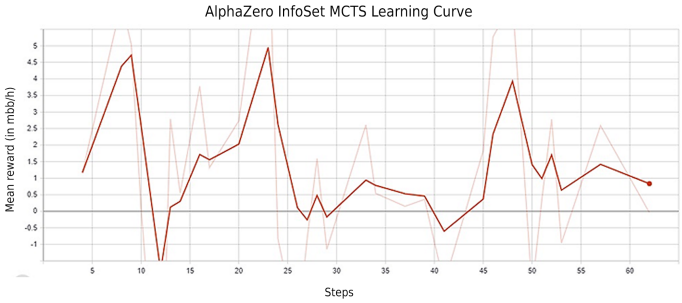

Fig. 4. Learning curve of the AlphaZero Infoset MCTS algorithm during training.

AlphaZero with Infoset MCTS. The AlphaZero Infoset MCTS agent had a positive reward against all agents but NFSP and Deep CFR. These results show that AlphaZero's policy and value networks possibly assist the plain Infoset MCTS search algorithm to take better actions but for more secure assumptions, a longer training process and a more thorough evaluation must be performed. As it is expected, however, the learning curve of the modified AlphaZero agent is does not seem to have an upward (learning) trend (Figure 4).

Neural Fictitious Self-Play (NFSP). The implemented NFSP agent was the best overall agent. It achieved the bigger mean rewards against all agents, and it also closely won against Deep CFR agent, which in theory is a better method than NFSP. The learning curve is shown in Figure 5.

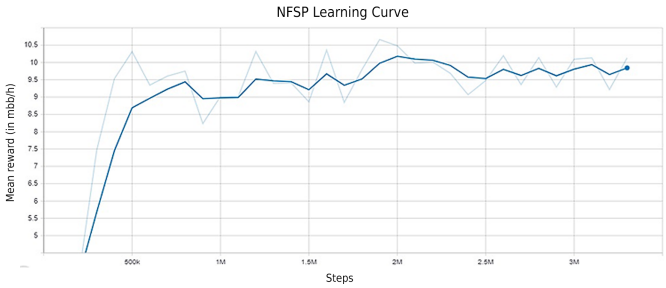

Fig. 5. Learning curve of the NFSP algorithm during training.

Deep CFR. The Deep CFR agent is the second-best agent, only behind NFSP. It achieved positive mean rewards against all agents except NFSP, from which it lost to a small margin. Furthermore, although the rewards were positive, they were smaller than NFSP's. While in theory Deep CFR overshadows

TABLE I
1-VERSUS-1 PERFORMANCE RESULTS IN MBB/H BETWEEN IMPLEMENTED AGENTS AFTER TRAINING.

	Random	Fish	Hand Evaluator	PPO	Infoset MCTS	AlphaZero Infoset MCTS	NFSP
Fish	250						
Hand Evaluator	591	2658					
PPO	1322	-136	2934				
Infoset MCTS	1653	936	-105	346			
AlphaZero Infoset MCTS	602	803	1605	3788	902		
NFSP	4616	3803	4611	4624	2918	1572	
Deep CFR	3106	3681	3106	1458	1126	1182	-304

NFSP, the computational limitations prevented Deep CFR's true potential. Due to the fact that the average policy training network is trained only at the end of the training, no learning curve was available, a fact that held back the efforts for finding the best possible training parameters.

LBR Evaluation. The results of the evaluation of HUNL agents against LBR are shown in Figure 6. The agent which is the least exploitable is Deep CFR. An interesting fact is that the second least exploitable agent is the one that always folds at the start of each game. While this agent never wins a game (an exception occurs when the other player folds first), it only loses 500 mbb when it starts the game first and has put the small blind, while it loses 1000 mbb when it plays second and has put the big bling. Although this agent presents very low exploitability, it has no use to anyone because it is not designed to win the game. All the other tested agents except Deep CFR have a higher exploitability because they bet bigger amounts of chips and thus are prone to losing more chips.

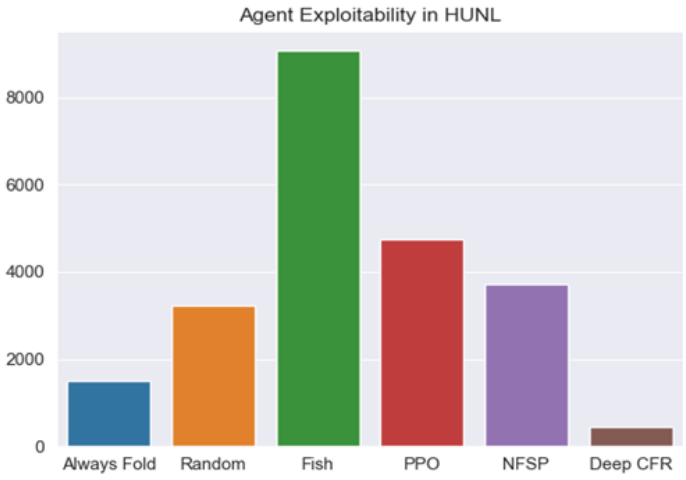

Fig. 6. Exploitability of implemented agents against LBR algorithm.

Another interesting fact is that the random agent is less exploitable than the agent with the best one-vs-one results, NFSP. This happens because this agent's strategy is uniformly random and has a higher probability of folding. This fact does not make the random agent a better agent than NFSP, but it only shows that it will lose the least amount of chips if it will play against a player that approximates Nash Equilibrium.

From the results of the one-vs-one evaluation and the evaluation against LBR of NFSP and Deep CFR agents one can assume that these agents have the best overall results. It is also obvious that although these agents are the best of all the others, they have a different approach and game style.

Figure 7 shows the number of actions each agent has performed in their one-vs-one evaluation. NFSP calls and raises half and full pot more than Deep CFR, which, on the other hand, folds, checks and performs all-in more.

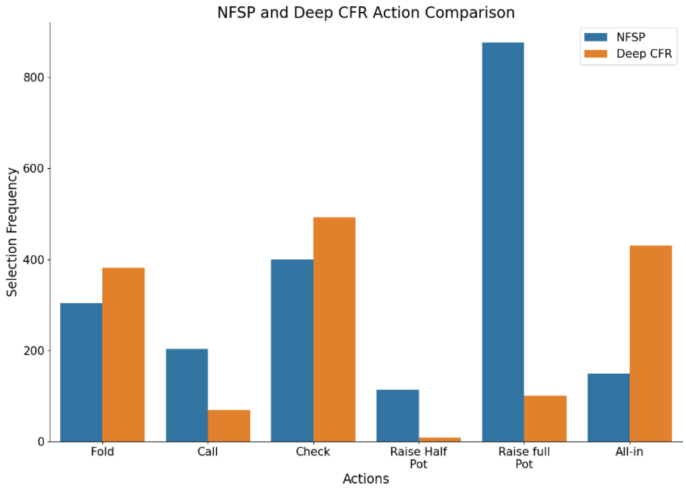

Fig. 7. Action Comparison Between NFSP and Deep CFR.

In general, the NFSP agent has a more aggressive style of play, and thus it has better results against easier opponents. On the other hand, Deep CFR plays more defensively at the first rounds and performs more all-ins when some or all board cards are dealt. Results from the evaluation against LBR show that NFSP's aggressive strategy makes it more exploitable to superior opponents, while Deep CFR is less exploitable even than the agent that always folds.

VI. ALPHABLUFF VIDEO GAME

To further test our agents in HUNL and provide a playground for human players against them, a Graphical User Interface (GUI) was created in the Unity game engine, which, along with the trained agents, constitutes AlphaBluff[2].

[2]https://ming.csd.auth.gr/alphabluff

In AlphaBluff, the player can choose between two modes and four game difficulties. The two available modes are Doyle's Game, which was previously presented, and Normal mode. The four game difficulties are: Easy, Normal, Hard and Very Hard. Each of these difficulties corresponds to "Fish", "Random", NFSP and Deep CFR agents respectively. Furthermore, the game keeps logs of every gameplay session, which includes wins and profits for every mode and difficulty.

A. Statistical Report

It is deemed necessary to see how AI agents perform against humans as well, thus, we gathered gameplay data of each AI player and computed their profit against each of the four gamestyles. In total, 429 game rounds were played (normal mode games were broken down to individual rounds) by 21 players of different levels of experience in Poker (Table II).

TABLE II
STATISTICAL ANALYSIS OF GAMEPLAYS BETWEEN AGENTS AND HUMANS.

	Easy (Fish)	Normal (Random)	Hard (NFSP)	Very Hard (Deep CFR)
Games played	80	122	136	154
Winrate	32%	49.1%	54.6%	72.8%

As expected, the "Fish" agent is quite an easy opponent for most human players, since its strategy is far from complex and can be easily beaten. Human players have a more difficult time against the Random agent, achieving wins about half of the time, however this is probably related to each player's level of experience. The NFSP agent has slightly more wins against human players (54.6%) than the Random agent, while the Deep CFR agent managed to win almost 3 out of 4 games played (72.8%). Considering the lack of computational resources for performing longer training sessions, these results are actually encouraging regarding their potential against human players.

VII. Discussion

Evaluation results showed promising indications that NFSP and Deep CFR can achieve significant performance in HUNL, outperforming all other agents. NFSP had the best overall results on the one-vs-one evaluations but its aggressive strategy made it struggle against a superior opponent. In contrast, Deep CFR had a very low exploitability, but its playstyle was not as rewarding as NFSP. Overall, performance of all agents would be further improved if there were no computational limitations.

Furthermore, evalution with LBR revealed interesting insights for each method and while it can be used as a measure of self-improvement, it cannot be used as a point of reference between different algorithms due to the stochastic nature of the game and the option of folding as an action in the game.

VIII. Conclusion

With AlphaBluff, we filled the gap between state-of-the-art research in AI and its applications in the field of gaming. More specifically, for this paper we trained various agents in the complex imperfect-information game Texas Hold'em Poker, we evaluated their performance and compared them against each other, which, to our knowledge, has never been carried out before. Additionally, a modern 3D GUI that was developed for the purposes of this work, which meets industry-level standards in regard to video game quality, constituting AlphaBluff, where human players can play against trained AI agents. This contributes to easy accessibility to AI models by the public, and also allowed us to perform an essential statistical analysis that was missing from such implementations.

References

[1] A. Lazaridis, A. Fachantidis, and I. Vlahavas, "Deep reinforcement learning: A state-of-the-art walkthrough," *Journal of Artificial Intelligence Research*, vol. 69, pp. 1421–1471, 2020.

[2] R. Aumann and S. Hart, "Handbook of game theory with economic applications," Elsevier, Tech. Rep., 2002.

[3] D. Billings, A. Davidson, J. Schaeffer *et al.*, "The challenge of poker," *Artificial Intelligence*, vol. 134, no. 1-2, pp. 201–240, 2002.

[4] M. Moravčík, M. Schmid, N. Burch *et al.*, "Deepstack: Expert-level artificial intelligence in heads-up no-limit poker," *Science*, 2017.

[5] M. Zinkevich, M. Johanson, M. Bowling *et al.*, "Regret minimization in games with incomplete information," *Advances in neural information processing systems*, vol. 20, 2007.

[6] N. Brown, T. Sandholm, and S. Machine, "Libratus: The superhuman ai for no-limit poker." in *IJCAI*, 2017, pp. 5226–5228.

[7] N. Brown and T. Sandholm, "Superhuman AI for heads-up no-limit poker: Libratus beats top professionals," *Science*, 2018.

[8] N. Brown, A. Bakhtin, A. Lerer *et al.*, "Combining deep reinforcement learning and search for imperfect-information games," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[9] E. Zhao, R. Yan, J. Li *et al.*, "Alphaholdem: High-performance artificial intelligence for heads-up no-limit texas hold'em from end-to-end reinforcement learning," in *AAAI*, 2022.

[10] H. Li, X. Wang, F. Jia *et al.*, "Rlcfr: Minimize counterfactual regret by deep reinforcement learning," *Expert Systems with Applications*, 2022.

[11] U. Berger, "Brown's original fictitious play," *Journal of Economic Theory*, vol. 135, no. 1, pp. 572–578, 2007.

[12] J. Heinrich, M. Lanctot, and D. Silver, "Fictitious self-play in extensive-form games," in *International conference on machine learning*, 2015.

[13] J. Heinrich and D. Silver, "Deep reinforcement learning from self-play in imperfect-information games," *arXiv preprint arXiv:1603.01121*, 2016.

[14] N. Brown, A. Lerer, S. Gross *et al.*, "Deep counterfactual regret minimization," in *International conference on machine learning*, 2019.

[15] D. Zha, K.-H. Lai, S. Huang *et al.*, "Rlcard: A platform for reinforcement learning in card games," in *International Joint Conference on Artificial Intelligence*, 2020.

[16] E. Liang, R. Liaw, R. Nishihara *et al.*, "RLlib: Abstractions for distributed reinforcement learning," in *International Conference on Machine Learning*, vol. 80, 2018, pp. 3053–3062.

[17] P. Moritz, R. Nishihara, S. Wang *et al.*, "Ray: A distributed framework for emerging {AI} applications," in *USENIX Symposium on Operating Systems Design and Implementation*, 2018, pp. 561–577.

[18] P. I. Cowling, E. J. Powley, and D. Whitehouse, "Information set monte carlo tree search," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 2, pp. 120–143, 2012.

[19] D. Silver, T. Hubert, J. Schrittwieser *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[20] M. Lanctot, E. Lockhart, J.-B. Lespiau *et al.*, "OpenSpiel: A framework for reinforcement learning in games," *CoRR*, vol. abs/1908.09453, 2019.

[21] T. Davis, N. Burch, and M. Bowling, "Using response functions to measure strategy strength," in *AAAI Conference on Artificial Intelligence*, 2014.

[22] V. Lisy and M. Bowling, "Equilibrium approximation quality of current no-limit poker bots," *arXiv preprint arXiv:1612.07547*, 2016.