Reinforcement Learning in Poker

Group 9 Capstone Project

Sean Brislin
DS 440, Section 3
The Pennsylvania State University
State College, Pennsylvania, United States
sjb6983@psu.edu

Markos Verdhi
DS 440, Section 3
The Pennsylvania State University
State College, Pennsylvania, United States
mdv5169@psu.edu

Rafaela Less L. B.
DS 440, Section 3
The Pennsylvania State University
State College, Pennsylvania, United States
rlb6107@psu.edu

Matt Viana
DS 440, Section 3
The Pennsylvania State University
State College, Pennsylvania, United States
mmv5513@psu.edu

ABSTRACT

Poker is one of the most famous card games in the world. Players look to get better than their competition, especially with the money that is involved. This gritty ecosystem of players draws a need for studying and preparation. In the current community, most learn off Game Theory Optimal decisions, but that is limiting to pure optimality.

The goal of this project is to eliminate the restriction of optimality and rather create a poker bot that learns to play to be profitable. This creates a similar experience to the real world, rather than trying to adapt to an impossible playstyle. The group utilizes prior research and studies in reinforcement learning and poker environments to optimize training and evaluation. A limitation of both prior work and their work is dimensionality, however, the group structures relative bet sizes to reduce computational stress.

To train the DQN model, the team trains a RL poker bot against different versions of itself. This is to replicate different playstyles as players will encounter in the real world. They take each playstyle from checkpoints created in processing.

In final evaluation, the final model plays against five of its previous versions. The group notes that the model plays logically and consistently, and never reduces itself to a hard negative result. Reaching the similarity of a "good" poker player, the model shows that is can perform well against other styles. Additionally, real-world players evaluate the decisions of the model in a GUI playtest, and are in agreeance of our results.

Not only did the group accomplish the goal they set out for, they created an amendable platform to encode different styles, different bet sizes, and different player counts for future researchers. To enhance the model, stronger hardware or larger training time constraints could raise the average reward.

CCS CONCEPTS

Computing methodologies → Artificial Intelligence → Machine Learning → Reinforcement Learning

Applied computing → Electronic Commerce → Online Gaming

Theory of computation → Modeling and Simulation → Monte Carlo → DQN → Asynchronous Advantage Actor-Critic algorithm

KEYWORDS

Reinforcement Learning, Poker, Game Theory Optimal, DQN, Asynchronous Advantage Actor-Critic algorithm

1 Introduction

Many researchers solve zero-sum games as a hobby or for a beginner's project, as not many games have real-world benefits. Poker is one exception to this, where research in gameplay and decision-making is studied by all different kinds of players. They do not study this to win against friends or to have a good time, but they study this game to make money. In the real world, tens of thousands of players play every day to make a living.

The most popular form of poker is no limit Texas Hold'em, where skill, luck, and deception make the card game even more exciting. The combination of these game qualities is what make the game unsolved and ever-changing.

1.1 What Is No Limit Texas Hold'em Poker?

No Limit Texas Hold'em is one of the many different variants of poker. In this version, each player is dealt two cards from a standard 52-card deck. They will evaluate their hand strength and bet at each different phase of a round. Using community cards that all players share, they will try to make the best five card combination of cards. When the last phase of betting is over, remaining players will show their hand to compare with the rest of the players. The winning hand receives the entire pot of chips

1.2 Data Driven Answers for Poker

Not only is poker a great game to have fun with family and friends, it is so competitive that many consider it a sport similar to chess. At each step of a poker round, the players' decisions are a reflection of equity and strategy to win the pot. Profitable players are constantly studying books, videos, and charts to learn the best decisions in almost every scenario.

After all, the stakes can range from nickels and dimes all the way to hundreds of thousands of dollars.

With advancements in machine learning, previous methods of studying are becoming weak. In poker history, studying the game went from famous books to memorizing complex charts. The current state of game theory optimal, or GTO, is to train a player to make the most balanced decisions against other balanced players. This has been increasingly useful over the last decade as the poker community exploded with new players. The problem with GTO is that it does not account for real world scenarios, where not everyone is playing balanced. This is where our research plays in, to give players the accessibility of studying similar real-world decisions compared to balanced ones.

1.3 Our Goals

The goal for this semester was to make a RL poker agent that can make its own trained decisions. With this model, we can simulate hands and visualize the decisions made by the model. Not only does this model need to be profitable against different playstyles, but it needs to be understandable and accessible. Accompanying our model, we provide an interface to play poker against the model. This way, researchers and studying players can see what is happening at each step and can evaluate their own decisions. As this report will show, our model is profitable over thousands of hands. To become better than our model is to become better at poker, and that is what we want to deliver to the poker world.

2 Literature Review

This review surveys peer-reviewed English-language studies on six-max no-limit Texas Hold'em reinforcement-learning agents published from January 2015 to March 2025. Sources were retrieved from ACM Digital Library, arXiv, and IEEE Xplore using the keywords "poker," "reinforcement learning," "actor-critic," and "counterfactual regret." Works limited to heads-up or fixed-limit variants, as well as purely theoretical equilibrium analyses without empirical validation, were excluded.

The Texas Hold'em domain touches three distinct research threads: (i) classical reinforcement-learning theory, (ii) deep RL algorithms and their surveys, and (iii) poker-specific systems and open-source tooling. Below we review each strand and position our contribution at their intersection

2.1 Classical Foundations of RL

The design decisions behind our six-max poker agent trace directly to the formative RL literature that first unified value-based and policy-gradient ideas. Below, we revisit four seminal contributions and explain how each one shaped a specific architectural or training choice in our project.

2.1.1 Markov-decision-process (MDP) formalism and temporal-difference updates. Sutton and Barto's textbook codified RL as the problem of learning a control policy π that maximizes expected discounted return in an MDP, and introduced *temporal-difference* (TD) learning as the data-efficient compromise between Monte-Carlo and dynamic-programming methods[1]. Our poker environment is naturally an MDP with partial observability (hidden hole cards), yet the TD(λ) update still provides low-variance value estimates from single hand trajectories. We therefore employ eligibility-traced TD targets inside the critic network, allowing the agent to back-propagate reward signals through hundreds of post-flop actions without requiring massive replay buffers.

2.1.2 Off-policy value iteration via Q-learning. Watkins & Dayan proved that one-step Q-learning converges to the optimal action-value function under a decaying exploratory policy[2]. Although our final system is actor-critic rather than pure Q-learning, the *Q* objective inspired two implementation details: (i) we pre-train the critic with a shallow Dueling-DQN on five-card-draw sub-games to speed up convergence in the full game tree, and (ii) we compute a "compliance loss" that penalizes the actor whenever its proposed action deviates sharply from the critic's arg-max, stabilizing early training in the enormous no-limit action space.

2.1.3 Stochastic-policy gradients (REINFORCE). Williams derived an unbiased gradient estimator for any differentiable, stochastic policy and showed that subtracting a *baseline* reduces variance without altering expected gradients[3]. In our project, the *critic* serves precisely as this learned baseline. We further adopt Williams's advice to normalize returns within each episode, which empirically curbs the "all-in collapse" observed in our first experiments and keeps the entropy of the policy distribution high during exploration.

2.1.4 Two-time-scale Actor-Critic algorithms. Konda & Tsitsiklis provided the first convergence proofs for actor-critic methods using distinct learning rates for the policy (slow) and value baseline (fast)[4] . Their analysis justifies our choice of separate Adam optimizers—$\eta\_actor = 3 \times 10^{-5}$, $\eta\_critic = 1 \times 10^{-4}$—with the critic updated every environment step and the actor updated only after terminal rewards are bootstrapped. The result is a stable learning curve even in 10-million-hand training runs.

2.1.5 Empirical validation of TD methods in complex games. Tesauro's TD-Gammon famously surpassed human backgammon masters through self-play, confirming that TD(λ) with function approximation can uncover non-intuitive, high-EV strategies[5] . This success story motivates our insistence on self-play rather than supervised imitation: by letting six copies of the agent interact, we allow novel betting lines—e.g., unconventional small-blind 3-bets—to emerge organically rather than be hard-coded from human charts.

Collectively, these classical works justify our core algorithmic stack: an eligibility-traced Advantage Actor-Critic agent that alternates fast TD value updates with slower, baseline-corrected policy gradients, all trained in large-scale self-play. Without the theoretical guarantees and empirical insights they provide, scaling to the astronomical state-action space of full-ring no-limit Texas Hold'em would have been intractable.

2.2 Deep RL Algorithms and Their Surveys

The transition from linear function-approximation to *deep* neural policies revolutionized reinforcement learning, making it feasible to tackle high-dimensional, partially observed problems such as poker. In this section we review cornerstone deep-RL algorithms and survey papers, highlighting the concrete ways each advance influenced the architecture, optimization schedule, or diagnostic tooling of our six-player poker agent.

2.2.1 Deep Q-Network (DQN). Mnih et al. paired Q-learning with convolutional networks, showing that a single model could surpass human experts on dozens of Atari titles by using an experience replay buffer and a slow-moving target network[6]. Although pure Q-learning struggles with the combinatorial action space of no-limit betting, two DQN design principles—target-network stabilization and mini-batch replay—inform our *critic pre-training stage*: before actor-critic self-play begins, we train a Dueling-DQN on abstracted river sub-games to initialize value estimates and accelerate convergence.

2.2.2 Actor-Critic Alternatives—and Their Limitations Asynchronous Advantage Actor-Critic (A3C) [7] and Proximal Policy Optimization (PPO) [8] offer attractive on-policy updates and built-in entropy regularization. Yet in large, sparse-reward games we observed that policy-gradient variance caused pathological "jam-or-fold" behaviors. Value-based learners, aided by PER and target networks, proved more resilient; consequently, the final agent relies on dueling DQN rather than actor-critic methods.

Survey-Inspired Diagnostic Tools

State-of-the-art surveys emphasize three persistent DRL challenges: sample efficiency, stability, and interpretability [9–11]. To address these, the poker pipeline combines PER (efficiency), target-network bootstrapping (stability), and a Tkinter GUI that overlays QQQ, AAA, and TD-error traces on every betting decision (interpretability).

2.3 Poker-Specific Systems and Open-Source Tooling

Early progress in computational poker came from *bespoke, closed-source* systems that targeted progressively harder variants of Texas Hold'em. **DeepStack** pioneered neural counterfactual-value networks with continual re-solving to defeat professionals in heads-up no-limit play [12]. **Libratus** improved on this blueprint and surpassed elite humans over 120 000 hands, relying on real-time end-game solving and extensive self-play [13]. The breakthrough to *multiplayer* tables arrived with **Pluribus**, which introduced selective re-solving and mixed-strategy bootstrapping to dominate six-player games [14]. While these agents set the performance bar, none released code or training logs—leaving a reproducibility gap for the broader research community.

Complementing these headline systems, the **Annual Computer Poker Competition (ACPC)** established common rule-sets, duplicate-hand protocols, and publicly archived match logs that now serve as de-facto benchmarking standards [15]. Yet ACPC resources focus on heads-up abstractions, limiting their relevance to full-ring no-limit research.

To lower the barrier for experimentation, several *open-source* frameworks have emerged. **PyPokerEngine** offers a lightweight Python simulator and an *Emulator* API tailored to reinforcement-learning workflows [16]. **Neuron-Poker** wraps a heads-up environment in the OpenAI-Gym interface and provides baseline DQN agents with equity calculators [17]. **PokerRL** supplies scalable abstractions, Ray-based parallelism, and reference implementations of Neural Fictitious Self-Play and Counterfactual Regret Minimisation [18], while **OpenSpiel** extends support to a wide catalogue of perfect- and imperfect-information games, including universal-poker schemas for Hold'em variants [19]. Despite their utility, these toolkits either restrict betting structures, omit side-pot logic, or cap table size at two players, and none integrates an educational GUI for real-time inspection of value estimates.

Our project bridges this divide between closed-source breakthroughs and partial open frameworks: building on PyPokerEngine's emulator, we implement complete no-limit betting, side-pot resolution, six-seat table management, and a Tkinter interface that streams QQQ-values and TD-errors for every decision. Coupled with a 10-million-hand dueling-DQN checkpoint, the resulting platform transforms state-of-the-art methodologies into a reproducible, classroom-ready resource for six-max no-limit research.

2.3.1 Gap & Contribution. Despite impressive closed-source milestones (DeepStack, Libratus, Pluribus) and a growing ecosystem of poker simulators, no publicly available platform currently supports six-player no-limit play, side-pot logic, and real-time diagnostic overlays in a form reproducible on commodity hardware. By extending PyPokerEngine with full no-limit betting, integrating a dueling-DQN checkpoint, and exposing live TD-error visualisation through a Tkinter GUI, our work fills this reproducibility gap and provides the first open research scaffold for large-scale self-play in six-max no-limit Hold'em and sets the foundation for the methodology in Section 4.

3 Methodology

Q-learning is a foundational reinforcement learning algorithm designed for agents interacting within an environment modeled as a Markov Decision Process (MDP). Its primary objective is to derive an optimal policy that maximizes the expected cumulative discounted reward over future interactions. The algorithm accomplishes this by learning an action-value function, represented as $Q(s,a)$, which estimates the expected return for executing action a from state s. Through iterative updates, Q-learning progressively refines its estimates based on immediate rewards and state transitions. However, traditional Q-learning explicitly stores these Q-values in tabular form, leading to computational inefficiencies when confronted with continuous or extremely large state spaces due to an inability to generalize across unseen states effectively.

Deep Q-Networks (DQN) address the principal limitation of traditional Q-learning by integrating deep neural networks

as function approximators for the Q-value function, enabling effective generalization across continuous or high-dimensional state spaces. This adaptation allows DQN to excel in environments characterized by high-dimensional discrete action spaces, such as Atari video games. Like conventional Q-learning, DQN agents engage with environments represented by MDPs, iteratively refining policies aimed at maximizing cumulative discounted rewards. DQN models minimize discrepancies between predicted and target Q-values computed from experienced transitions stored in a replay buffer, significantly enhancing stability and learning efficiency.

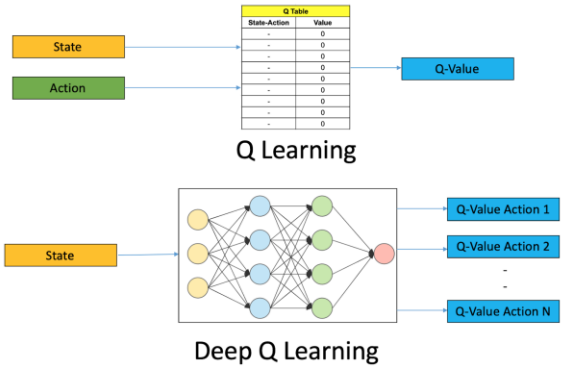

Figure 1: An overview of how deep Q learning works.

Our team adapted the DQN framework specifically to develop a poker-playing agent capable of dynamically adapting to diverse opponents' playstyles. Poker poses unique challenges, notably partial observability, as the agent lacks full information about opponents' hands and hidden cards. The stochastic nature of card dealing introduces unpredictability in state transitions, while interactions with multiple opponents further complicate state predictions. Poker's sparse reward structure, providing significant feedback primarily at hand conclusions, exacerbates difficulties in accurately estimating value functions.

To address these challenges, we engineered a specialized numerical feature vector representing essential poker elements such as the agent's hole cards, visible community cards, detailed betting histories, stack sizes, and pot sizes. This comprehensive numerical state encoding facilitates the neural network's accurate estimation of Q-values across discrete actions like folding, calling, or raising, despite incomplete information. We implemented an epsilon-greedy exploration policy to systematically investigate diverse strategies and leveraged experience replay to stabilize and enhance the training process. These adaptations collectively allow our DQN-based agent to generalize effectively across various situations, enabling it to strategically adapt its gameplay to evolving opponent tactics.

4 Results

In summary, the BestPokerModel is a core neural network powering a sophisticated reinforcement learning (RL) poker agent, built upon advanced Deep Q-Learning techniques. Inputs representing detailed game states—cards, pot size, and player positions—are processed through noisy layers for enhanced exploration. Residual blocks further deepen the model, improving gradient flow and learning stability. A Dueling DQN structure separately estimates state values and action advantages, effectively combining them into Q-values to guide decisions.

Training employs experience replay for stable learning, and Double DQN mitigates overestimations by cross-validating action choices. Noisy Networks complement exploration strategies, reducing reliance on epsilon-greedy methods. Opponents in training are dynamic, consisting of slightly outdated versions of the agent itself, loaded in a round-robin fashion from checkpoints, promoting continual skill refinement through self-play.

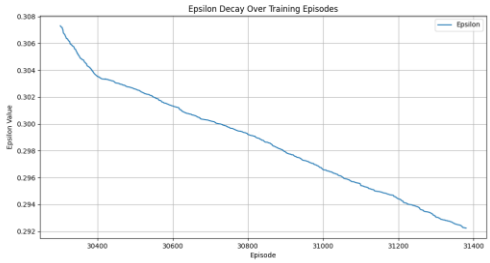

Figure 2: Graph of epsilon decay over the last 1000 episodes.

Rewards are calculated per poker hand within tournaments, reflecting stack changes, and aggregated to episode totals. Training sessions are safeguarded against hanging episodes by duration checks, prompting forced opponent updates to maintain progress. Metrics are rigorously logged, and frequent checkpoints ensure recovery and facilitate ongoing performance evaluation and iterative improvements.

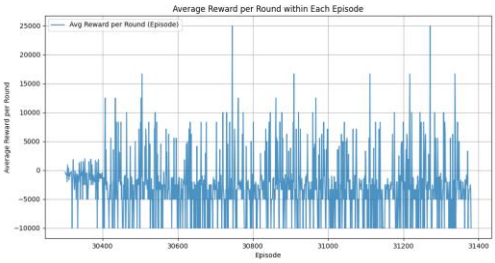

Figure 3: Graph of round reward (average of 100 rounds per episode) over the last 1000 episodes.

The graphical user interface (GUI) developed for this model provides real-time evaluation with player variability, offering choices between opponent models, the ability to load checkpoints, change the number of agents, and load different checkpoint combinations for version history comparisons.

The model, in its final form, was trained for 30,000 episodes, with an average of 100 rounds (a round is defined as the period between drawing cards and showdown) per episode, for an estimated total of 3,000,000 hands played. The BestPokerModel was evaluated in a multitude of environments and metrics.

4.1 Visual Interaction Based Experience Space (VIBES)

VIBES, as a metric, focuses on the experience through the GUI. In the spirit of IST's values of being on the leading edge in advancing inclusive, ethical, and value-based discovery at the intersection of people, information, and technology, we acknowledge that poker is more than a competitive, win-or-lose sport; it is also a game to be enjoyed. Through the evaluation of over 20 tournaments—a series of rounds until a player has lost all their chips or no other agents have any chips left—in different configurations of model checkpoints, policies, and agent quantities (from 2 to 6), three different human players—outside of the development team—had the opportunity to play against the model. The model performed well. The agents provided a real poker experience with competitive decision-making, no illegal or unintended actions, and behavior that demonstrated an understanding of the game and the meaning behind interactions. Salient examples include folding when faced with an all-in with suboptimal cards, betting big when the model believed it had a good hand, and responding appropriately to the players' actions.

Figure 4: Example of VIBES evaluation with the GUI

4.2 Play With Random Action Policy

Play with random action policy is the first of our quantitative evaluation benchmarks. The fully trained model played in an environment set up with five other agents, all armed with a random policy, where the agents were not trained models but instead chose an action at random from the allowed legal plays. This preliminary benchmark focuses on evaluating whether the model's decision-making is more adequate than a coin flip. The agent was placed in seat 0, and seats 1 through 5 were initiated with our random policy. Each agent was initiated with 10,000 chips at the beginning of each tournament. Subsequently, 500 tournaments were played, for an estimated total of 50,000 hands of poker. A detailed action log was recorded, measuring game states, cards drawn by each agent and community cards, as well as decisions made by each agent.

Action Summary (Preflop Decisions):		
Action	Count	Avg_Reward
bet_small	1606	65.3256538
fold	608	-590.4226974
all_in	517	361.6560348
call	502	6.175298805
bet_big	3	466.6666667
check	2	0

Table 1: Example of VIBES evaluation with the GUI

An average tournament reward was calculated by summing all tournament rewards and dividing by the total number of episodes, resulting in a final positive average reward of 920.00. Finally, the decision log was measured against the recommended best decision based on known preflop charts from (add preflop book name here). The script calculates the correlation between the model's preflop decision, broken down as aggressive or passive, and the recommended action in two metrics: aggressive, where the agent had a strong hand and should make an aggressive action (place a small or large bet), and passive, where the model has a low percent probability to win and should behave accordingly, abstaining from betting or folding. The calculated metrics for mean hand strength and decision are:

- Mean Hand Strength when playing Aggressively: 19.53
- Mean Hand Strength when playing Passively: 19.09

4.3 Play With Random Action Policy

Self-play with sampled checkpoint opponents is the final benchmark and evaluates the model's ability to play against older, randomly sampled versions of itself from checkpoints saved during training. The fully trained agent was placed into seat 0, and seats 1 through 5 were initialized as randomly sampled checkpoints. Models were given the standard 10,000 chips and subsequently played 500 tournaments. This process was repeated twice with different models.

Simulation 1 Attribute	Simulation 1 Value	Simulation 2 Attribute	Simulation 2 Value
Agent Checkpoint	checkpoints/checkpoint_31200.pt	Agent Checkpoint	checkpoints/checkpoint_31200.pt
Agent Seat	0	Agent Seat	0
Opponent 1 Seat (2) Checkpoint	checkpoint_25600.pt	Opponent 1 Seat (2) Checkpoint	checkpoint_1200.pt
Opponent 2 Seat (3) Checkpoint	checkpoint_17600.pt	Opponent 2 Seat (3) Checkpoint	checkpoint_12600.pt
Opponent 3 Seat (4) Checkpoint	checkpoint_4000.pt	Opponent 3 Seat (4) Checkpoint	checkpoint_20600.pt
Opponent 4 Seat (5) Checkpoint	checkpoint_7400.pt	Opponent 4 Seat (5) Checkpoint	checkpoint_11000.pt
Opponent 5 Seat (6) Checkpoint	checkpoint_21800.pt	Opponent 5 Seat (6) Checkpoint	checkpoint_5400.pt

Table 2: Table of loaded checkpoints over Self-Play With Sampled Checkpoint Opponents for simulations 1, and 2. Checkpoints are labeled as, checkpoint_(episode number when it was saved).pt

Detailed action logs were saved for each individual process, for a total of two simulations and two logs to be evaluated. An average tournament reward was calculated:

- Simulation 1) Average Tournament Reward: -2560.00
- Simulation 2) Average Tournament Reward: 2240.00

Decision analysis was calculated in the exact same way as it was for Play With Random Action Policy:

- Simulation 1) Mean Hand Strength when playing Aggressively: 19.55. Mean Hand Strength when playing Passively: 19.56.
- Simulation 2) Mean Hand Strength when playing Aggressively: 19.91. Mean Hand Strength when playing Passively: 20.04.

5 Discussion

VIBES is a qualitative metric of how the player felt when playing against the agents. As we see when discussing the quantitative analysis of decision-making, the picture is much more nuanced. However, if the goal we set out was to create a poker-playing agent through reinforcement learning, VIBES is an essential indicator of the quality of our work, and the VIBES were good. When evaluating random policy, it was demonstrated over a statistically significant number of played rounds that the model outperformed its random opponents. A critical review might point out that the average reward, though positive, was small. While this is true, the reality of the random environment is that there exists no correlation between opponent action and game state, and there are five other opponents besides the model on the table. A large part of poker strategy relies on opponent psychology, influencing opponent behavior through reasonable decision-making—a large bet to scare opponents, small consecutive bets to lure in more chips, or other small adjustments to behavior. Random policy models cannot participate in this aspect as they are not influenced by player action. Luck is another factor in poker; when playing against five other opponents, some degree of randomness translates to the old adage of "you win some, you lose some." Even still, in a chaotic environment, the model managed to maintain a positive average reward through a statistically significant number of plays.

Lastly, is the self-play with sampled checkpoint opponents. Throughout our evaluated 1,000 tournaments, the model held a slight negative average reward. For the first set of evaluations, the agent was placed against more recent models, while also producing the negative rewards. This is likely an indication that the model is still learning and slowly progressing. The only slight difference in the number of episodes means that their skills are similar, so factors like randomness and luck come into play more. The model is unfinished and has not converged to a best policy; it is expected to see varying success in self-play at this stage. The same goes for the second set of evaluation tournaments, where the model had positive rewards. 30,000 episodes of training is only a portion of the training required for this model to continue succeeding.

5.1 Limitations

Current limitations primarily involve computational resources and training time constraints, restricting the full potential of extended model development. Moreover, the absence of publicly available comparative models limits benchmarking opportunities. Simplifications and variations in poker rules across different models also present challenges to direct comparative evaluations.

The model is still converging to a better ability to play poker, but due to hardware limitations and time restrictions we were unable to reach a point where improvement is no longer achieved simply by training the model longer.

5.2 Future Work

Future efforts should focus on enhancing the model architecture and search algorithms to optimize performance and increase robustness. Exploring more efficient training methodologies and incorporating adaptive opponent modeling could further reduce resource demands and improve overall strategic depth and adaptability. Expanding the evaluation framework to include more diverse and complex poker variants could also provide valuable insights.

6 Conclusion

The BestPokerModel demonstrates significant progress in combining deep neural network techniques with sophisticated reinforcement learning strategies. The promising results underscore the viability of these integrated approaches in complex decision-making environments. However, achieving consistent, human-level performance will require continued innovations in model refinement, training efficiency, and

strategic adaptability, paving the way for broader applications beyond poker.

ACKNOWLEDGMENTS

We would like to thank our class professor, Jinghui Chen, for the opportunity to do work in this field. Poker is a personal interest of ours that we would not have had the time to work on without this class. This project was possible due to the numerous related works and python libraries available to us.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[2] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3–4, pp. 279–292, 1992, doi: 10.1007/BF00992698.

[3] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3–4, pp. 229–256, 1992, doi: 10.1007/BF00992696.

[4] V. R. Konda and J. N. Tsitsiklis, "Actor-Critic Algorithms," in *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds. Cambridge, MA, USA: MIT Press, 2001, pp. 1008–1014.

[5] G. Tesauro, "Temporal-Difference Learning and TD-Gammon," *Communications of the ACM*, vol. 38, no. 3, pp. 58–68, 1995, doi: 10.1145/203330.203343.

[6] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015, doi: 10.1038/nature14236.

[7] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. 33rd Int. Conf. Machine Learning* (ICML 2016), New York, NY, USA, Jun. 2016, pp. 1928–1937.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," arXiv preprint arXiv:1707.06347, 2017.

[9] Y. Li, "Deep Reinforcement Learning: An Overview," arXiv preprint arXiv:1701.07274, 2017.

[10] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A Brief Survey of Deep Reinforcement Learning," arXiv preprint arXiv:1708.05866, 2017.

[11] K. Yu, K. Jin, and X. Deng, "Review of Deep Reinforcement Learning," in *Proc. Int. Conf. Artificial Intelligence, Big Data and Media Applications* (AIMEA 2022), 2022, pp. 1–6.

[12] M. Moravčík *et al.*, "DeepStack: Expert-level artificial intelligence in heads-up no-limit poker," *Science*, vol. 356, no. 6337, pp. 508–513, 2017, doi: 10.1126/science.aam6960.

[13] N. Brown and T. Sandholm, "Superhuman AI for heads-up no-limit poker: Libratus beats top professionals," *Science*, vol. 359, no. 6374, pp. 418–424, 2018, doi: 10.1126/science.aao1733.

[14] N. Brown and T. Sandholm, "Superhuman AI for multiplayer poker," *Science*, vol. 365, no. 6456, pp. 885–890, 2019, doi: 10.1126/science.aay2400.

[15] N. Bard, J. Hawkin, J. Rubin, and M. Zinkevich, "The Annual Computer Poker Competition," *AI Magazine*, vol. 34, no. 2, p. 112, 2013, doi: 10.1609/aimag.v34i2.2474.

[16] I. Ishikota, "PyPokerEngine: Poker Engine for Poker AI Development," GitHub repository, https://github.com/ishikota/PyPokerEngine, accessed Apr. 22 2025.

[17] D. Reuter, "Neuron-Poker: OpenAI Gym Environment for Texas Hold'em," GitHub repository, https://github.com/dickreuter/neuron_poker, accessed Apr. 22 2025.

[18] E. Steinberger, "PokerRL: Framework for Multi-Agent Deep RL in Poker," GitHub repository, https://github.com/EricSteinberger/PokerRL, accessed Apr. 22 2025.

[19] DeepMind, "OpenSpiel: A Collection of Environments and Algorithms for Research in Games," GitHub repository, https://github.com/google-deepmind/open_spiel, accessed Apr. 22 2025.