Universiteit Leiden
The Netherlands

DSAI

Bluffing tendencies in Leduc Hold'em

Tarik Začiragić

Supervisors:
Prof.dr. Aske Plaat
Prof.dr. Joost Batenburg

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

01/07/2025

In the game of poker, bluffing is an essential skill. When humans play poker, they bluff. In this thesis we looked at whether DQN and CFR agents exhibit bluffing behavior in Leduc Hold'em when they are trained on each other simultaneously. Leduc Hold'em is a simplified version of poker. Most works on poker focus on performance metrics such as win rates, while bluffing is overlooked. We designed an experiment where we let the DQN and CFR agent play against each other while we log their actions and all environment variables to use for the bluff analysis. We have found that both DQN and CFR exhibit bluffing behavior. Both of them attempt to perform a bluff, however the percentage of successful bluffs (where the opponent folds) is much smaller. In future work, this research should be expanded to the full game of poker.

Contents

Chapter 1

Introduction

1.1 Topic

Bluffing is a fundamental aspect of strategic behavior in many imperfect-information games. In poker, a popular example of an imperfect-information game, players routinely make decisions based not only on the strength of their hand, but also on the perceptions and potential reactions of their opponents. Successfully executing a bluff involves both masking weakness and exploiting uncertainty which are skills that are traditionally associated with human intuition and psychological reasoning [vNM44]. However, with recent advances in artificial intelligence and reinforcement learning, the question arises: Can algorithmic agents also learn to bluff?

This thesis investigates the emergence and characteristics of bluffing behavior in two widely studied AI algorithms, namely, Deep Q-Networks (DQN) [MKS+15] and Counterfactual Regret Minimization (CFR) [ZJBP07], within the simplified poker environment of Leduc Hold'em.

Modern reinforcement learning agents have demonstrated the capacity to learn strategic behaviors in complex environments with imperfect information, including aspects of deception and opponent modeling, as shown by AlphaStar's success in StarCraft II [VBC+19]. The core objective of this research is to understand whether and how DQN and CFR agents learn to bluff, how their bluffing styles differ, and how each agent reacts when bluffed. To this end, agents were trained under various conditions, namely, against random opponents, against frozen versions of each other, and in simultaneous co-training scenarios. Their behaviors were then analyzed using a custom bluff detection heuristic and metrics which were derived from the logged game data.

By studying bluffing tendencies at both the action and reaction level, this thesis contributes to a deeper understanding of how artificial agents navigate uncertainty and deception in adversarial settings, building on recent interest in modeling human-like strategic behavior in machines [BS19].

1.2 Problem Statement

The problem statement of this thesis is:

- Do algorithmic agents exhibit bluffing tendencies in Leduc Hold'em?

Specifically, this problem statement can be broken down into 3 research questions that are going to be answered in the rest of this thesis:

- RQ1: How should CFR & DQN agents be trained?
  - This question investigates in what way the CFR and DQN agents should be trained to have strong agents?
- RQ2: Do CFR and DQN bluff?
  - This question focuses on identifying and quantifying bluffing behaviors in each agent.
- RQ3: What are the agents reactions to bluffing?
  - This question examines how each agent responds when confronted with a potential bluff from the opponent.

1.3 Methods

In order to answer the problem statement from section 1.2, we are going to perform empirical research in which we will extract certain metrics that will both be described and explained in Chapter 4. These should give us valuable insight into the topic.

1.4 Contribution

The key contributions of this thesis are:

- We show that DQN can be cross trained on CFR and vice versa.
- We show that both CFR and DQN bluff in Leduc Hold'em.
- We show how algorithmic agents react to bluffing by opponents in Leduc Hold'em.

These contributions help illuminate how bluffing emerges in Leduc Hold'em due to both the algorithms and the rules of the game.

1.5 Reading guide

The reminder of the thesis is structured as follows:

Chapter 2 provides the relevant background on Leduc Hold'em and the learning algorithms used in this work, namely Deep Q-Networks (DQN) and Counterfactual Regret Minimization (CFR). Furthermore, the concept of bluffing in strategic decision-making is also introduced.

Chapter 3 reviews related work on bluffing behavior in artificial agents, covering research in poker and other bluff-centric domains such as the game Coup.

Chapter 4 describes the experimental methodology that is used to train and evaluate the agents. This includes the setup of the RLCard environment, training procedures for both DQN and CFR agents, the logging and analysis tools used to extract the bluffing behavior, and the heuristic definitions that are adopted for identifying bluffing.

Chapter 5 provides the results from the experiments and an interpretation of them. Most importantly, it also provides answers to the research questions, and therefore, the problem statement through deeper analysis of the results.

Chapter 6 provides a conclusion to the thesis while also indicating the limitations and possible future work.

Chapter 2

Definitions

This chapter introduces key concepts and terminology that are fundamental to understanding the discussions and results, which will follow in the subsequent chapters.

2.1 Achievements in CFR, DQN and AI Poker

Over the years, significant progress has been made in the application of AI to strategic games, and in particular, to poker. Counterfactual Regret Minimization (CFR) has been a foundational algorithm for solving imperfect-information games and has shown remarkable success. In 2017, DeepStack introduced a novel approach by combining CFR with deep learning, using neural networks to estimate counterfactual values in future game states during real-time play. This depth-limited continual re-solving strategy allowed DeepStack to reach expert-level performance in heads-up no-limit Texas Hold'em, demonstrating the feasibility of integrating machine learning with game-theoretic reasoning [MSB+17]. In 2019, Pluribus, an AI system built on CFR variants and abstraction techniques, achieved superhuman performance in six-player no-limit Texas Hold'em, marking a milestone in multiplayer game AI [BS19]. It consistently defeated professional poker players without relying on prior human data or strategies.

Simultaneously, Deep Q-Networks (DQN) revolutionized the field of reinforcement learning by combining deep learning with Q-learning. Introduced by Mnih et al. (2015), DQN achieved human-level control on Atari 2600 games, proving that deep neural networks could learn effective policies directly from high-dimensional inputs [MKS+15].

Beyond poker, multi-agent reinforcement learning has also demonstrated superhuman achievements. In 2019, DeepMind's AlphaStar reached grandmaster level in StarCraft II, a complex real-time strategy game, showcasing the ability of cooperative and competitive RL agents to master long-term planning in highly stochastic environments [VBC+19].

These milestones highlight the capability of algorithmic agents to match, and in many cases, to surpass human expertise in strategically complex domains.

2.2 Leduc Hold'em

(Limit) Leduc Hold'em is a simplified variant of poker which is commonly used for game theory research because of its small state space and action space. It was first introduced as a simplified poker environment in the 2005 paper "Bayes' Bluff: Opponent Modeling in Poker" [SBL+05]. It was designed to retain the core strategic features of poker while being computationally tractable. This makes it a valuable testbed for research in imperfect-information games.

The game of Leduc Hold'em is played with a small deck which consists of 6 cards, namely, 2 copies of each of the 3 distinct ranks (denoted as King, Queen and Jack). There are 2 players who compete over 2 betting rounds. At the start of each game, both players are dealt 1 private card, which is not shown to their opponent. After the private cards are dealt, the first betting round occurs where players can choose to either call, raise, or fold during their turn. The betting follows a fixed limited structure (hence Limit Leduc Hold'em), where the number of raises per round is capped, and the bet sizes are predetermined. However, it is to be noted that no-limit variants exist where there are no caps nor restrictions on the betting amounts.

Following the first betting round, a single community card is revealed face-up on the table. A second and final betting round then takes place, with the same betting options available to the players. If neither player folds during the course of the game, a showdown occurs at the end of the 2nd betting round. The winner is determined by the best hand, according to the following hierarchy:

- A pair (2 cards of the same rank) beat a single high card.
- If neither player has a pair, the player with the higher private card wins.

The pot is awarded to the winning player. In the case of a fold, the player who did not fold immediately wins the pot, without needing to reveal their card.

Despite its simplicity, Leduc Hold'em retains several core strategic features of full-scale poker, namely:

- Imperfect information: Players do not know their opponent's private card.
- Hidden strength: The strength of a hand can change significantly once the public card is revealed.
- Bluffing and deception: Even though it has a smaller state and action space, it is large enough to allow for potential strategic play and deception.

For this thesis, we use the implementation of Leduc Hold'em provided by RLCard [ZLC+19] which is described in Chapter 4.

2.3 Bluffing

Bluffing is one of the most iconic aspects of poker. It involves a player intentionally misrepresenting the strength of their hand by typically betting or raising with weak hands in order to convince

opponents to fold their stronger hands. The effectiveness of a bluff relies on the opponents perception of the players behavior, previous actions, betting patterns, and psychology. The goal is not just to win individual hands, but to remain unreadable over time [Skl87].

A successful bluff relies not only on bold action, but also on an understanding of how ones behavior may be perceived by others. As such, bluffing is often regarded as a test of psychological insight, timing and the ability to manipulate expectations under uncertainty [Skl87].

In regular real-life poker played between humans, players often rely on reading behavioral cues such as body language, facial expressions, gaze, etc. Skilled players may exploit opponents who bluff too frequently or not enough. In contrast, AI agents operating under purely algorithmic strategies bluff as a byproduct of optimal play, rather than psychological manipulation.

From a game-theoretic perspective, bluffing is essential for maintaining an unpredictable strategy and ensuring that a player is difficult to exploit. Bluffing is needed to construct a mixed strategy that is part of a Nash equilibrium. At equilibrium, players randomize their actions such that their opponents are indifferent to which choice they make. For example, if a player only bets with strong hands, they become exploitable as the opponents will simply fold when they bet and call when they check which will break the balance and allow for exploitation. By bluffing appropriately, a player makes it harder for opponents to infer their hand strength, which in turn forces them to make a decision under uncertainty, and this preserves optimal play [CA06].

2.4 Counterfactual Regret Minimization (CFR)

Counterfactual Regret Minimization (CFR) is a prominent iterative algorithm for computing approximate Nash equilibria in extensive-form games with imperfect information such as poker [ZJBP07]. In such games, players do not have full visibility of the game state, which makes it necessary to reason over information sets which are collections of decision nodes that are indistinguishable from the players perspective (decision points where a player cannot distinguish between multiple game states because of hidden information).

CFR works by simulating repeated plays of the game. CFR tracks for each decision point in the game (information set), how much a player "regrets" not having chosen each available action in the past. These regrets are computed using counterfactual values which are expected outcomes assuming the player had taken a different action, while the rest of the game proceeds as before [ZJBP07].

In each iteration, the algorithm performs a traversal of the game tree, during which it:

1. Computes the counterfactual value of each action at each information set.
2. Updates the cumulative regret for not having taken each action.
3. Updates the strategy at each information set using regret matching which is a rule that increases the probability of choosing actions with higher positive cumulative regret.

The strategy at each information set is updated based on cumulative regrets using the regret matching rule:

- Actions with higher positive cumulative regret are assigned proportionally higher probability.
- If all regrets are non-positive, the algorithm defaults to a uniform strategy.

To improve convergence, CFR maintains an average strategy over iterations, rather than using the strategy from a single iteration [ZJBP07].

2.5 Deep Q-Networks (DQN)

Deep Q-Networks (DQN) are a class of model-free reinforcement learning algorithms that approximate the optimal action-value function using deep neural networks [SB18]. It is a combination of the theoretical framework of Q-Learning and the function approximation capabilities of deep learning [MKS+15]. DQN has achieved breakthrough success in Atari video games and in continuous action spaces [HMVH+18].

Q-Learning aims to learn the optimal Q-function $Q^*(s, a)$ that estimates the expected cumulative reward of taking an action a in state s and following the optimal policy π^* thereafter [WD92]. However, the issue with tabular Q-learning is scalability to environments with large or continuous state spaces. To handle large or complex state spaces, DQN replaces the Q-table from tabular Q-learning with a neural network that estimates the Q-value function $Q(s, a; \theta)$, where θ represents the parameters (weights and biases) of the neural network [MKS+15]. This network takes a state s as input and outputs an estimated Q-value for each possible action. By using a neural network the agent can generalize from previously seen states to new ones, without having to learn a separate value for every possible state-action pair.

The key components of DQN are:

- Q-Network (Function Approximator): The Q-network takes a state s as input and outputs Q-values for all possible actions. The network is trained to minimize the temporal difference (TD) error which can be formally defined as

$$L(\theta) = E_{(s,a,r,s')}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\right)^2\right] \tag{2.1}$$

  where

  - θ are the parameters (weights and biases) of the main Q-network,
  - θ^- are the parameters of a separate target Q-network which is updated less frequently for stability.
  - $\gamma \in [0, 1]$ is the discount factor,
  - r is the immediate reward,
  - s' and a' are the next state and next action respectively, and

  - $\max_{a'} Q(s', a'; \theta^-)$ represent the estimated optimal future value.

- Experience Replay: To break the correlation between sequential observations, DQN uses a replay buffer to store past experiences (s, a, r, s'). During training, minibatches of past experiences are sampled from this buffer to update the Q-network. When an agent interacts with the environment, the interactions are sequential and highly correlated to each other as each state depends on the previous one. Training a neural network on such correlated data can lead to unstable learning and poor generalization. Thus, to address this, DQN uses a replay buffer from which minibatches of transitions are sampled uniformly at random which causes the temporal correlations between consecutive experiences to break and the Q-network learns more robust value estimates [MKS+15].

- Target Network: A separate target network $Q(s, a, \theta^-)$ is used to compute the target Q-values in the loss function. Its weights θ^- are periodically updated with the main networks weights θ. This helps to reduce the instability that arises from using rapidly fluctuating Q-value estimates during training [MKS+15].

The general training procedure is as follows:

1. The agent interacts with the environment, collects transitions (s, a, r, s') and stores them in the replay buffer.
2. At each training step, a minibatch of transitions is sampled.
3. The target Q-values are computed using the target network.
4. The main Q-network is updated by minimizing the TD loss.
5. Periodically, the targets networks parameters are synchronized with the main network.

2.6 Summary of Background

In this chapter, we have introduced the foundational concepts which are necessary to understand the research conducted in this thesis. We began by providing a brief overview of major AI achievements in poker. Then we outlined the structure and strategic characteristics of Leduc Hold'em. Further, we defined bluffing in the context of traditional poker. Lastly, we described the basic mechanics of the CFR and DQN algorithms.

Chapter 3

Related Work

Bluffing, as a form of strategic deception in imperfect-information games, has long served as a benchmark for intelligent behavior in artificial agents. However, most research in the domain focuses on performance outcomes or equilibrium convergence, rather than on dissecting strategic behavior like bluffing itself.

To date, no prior work has directly compared nor analyzed the bluffing behavior of agents trained with CFR (Counterfactual Regret Minimization) and DQN (Deep Q-Networks) in the Leduc Hold'em environment. Even in larger-scale domains like Texas Hold'em, existing studies have emphasized win rates and exploitability, but not the qualitative nature of strategies such as bluffing or deception. This thesis addresses that gap by providing a comparative analysis of bluffing strategies in CFR and DQN agents. The goal is not just to measure performance, but to interpret whether and how bluffing manifests itself in these fundamentally different algorithmic approaches.

A significant contribution to the analysis of bluffing behavior in poker comes from Southey et al. in Bayes Bluff: Opponent Modeling in Poker (2005). The authors propose a Bayesian probabilistic model that separates the uncertainty of the game dynamic from the uncertainty in an opponents strategy. Their framework enables inference over opponent strategies based on observed actions, allowing the agent to compute a posterior distribution over possible opponent policies. While it is not explicitly focused on bluff detection and analysis, their work introduces several methods (Bayesian Best Response, MAP response, and Thomspon sampling) that adapt agent behavior based on observed opponent tendencies. By testing these methods in both Leduc and Texas Hold'em, they demonstrate that even with limited data, Bayesian agents can learn to exploit opponents effectively including learning to respond to deceptive strategies like bluffing. Their use of both independent Dirichlet priors and informed priors based on expert knowledge provides a strong precedent for modeling strategy uncertainty, which closely aligns with the goals of analyzing and interpreting bluffing behavior in learned agents [SBL+05].

Furthermore, recent advancements in artificial intelligence have explored the modeling of bluffing behavior in social deduction games like Coup. Guo et al. (2023) introduced Suspicion-Agent which is an agent that is built on the GPT-4 model and it designed to model theory of mind (ToM) in strategic gameplay. Without explicit training on the rules of Coup, the agent was able to infer hidden information, anticipate opponent beliefs, and employ bluffing strategies by claiming character

roles it did not possess. This work provides a novel perspective on bluffing behavior in a setting where social reasoning and belief manipulation are important [GYY+23].

Additionally, benchmark systems like DeepStack (Moravčík et al., 2017) and Libratus (Brown & Sandholm, 2019) showcased that AI can achieve superhuman performance in heads-up no-limit Texas Hold'em. While DeepStack pioneered real-time solving with neural network value functions [MSB+17], Libratus proved capable of long-form strategic dominance using a game-theoretic approach without learning from neural approximations [BS19]. Both systems implicitly exhibit bluffing behavior though their focus is on minimizing exploitability.

Lastly, a recent and important contribution to solving imperfect-information games is ReBeL (Recursive Belief-based Learning) by Brown et al. (2020). ReBeL presents a unified framework that combines reinforcement learning and search in both training and evaluation, achieving superhuman performance in heads-up no-limit Texas Hold'em. The core innovation of ReBeL is the introduction of public belief states (PBSs), which represent the common knowledge distribution over game states, allowing the use of search and value estimation in imperfect-information environments. ReBeL trains a policy and value network over these PBSs using self-play and uses CFR-based search in subgames to approximate Nash equilibria. ReBeLs contribution lies in generalizing the AlphaZero-style RL+Search paradigm to imperfect-information games [BBLG20].

To conclude, this chapter has focused on describing some prominent existing related works which paved the way in the fields of bluffing analysis and AI poker.

Chapter 4

Methodology

This chapter outlines the methodology used to investigate bluffing behavior in algorithmic agents within the Leduc Hold'em environment. It will provide details of the training setup, agent configurations, evaluation procedures and the custom heuristic that was developed to detect and analyze bluffing. By logging agent decisions we aim to uncover how bluffing emerges and manifests itself in both the CFR and DQN agents.

4.1 Environment Setup

All experiments were conducted using the Leduc Hold'em environment provided by the Python RLCard framework [ZLC+19]. Leduc Hold'em, as explained before, is a simplified poker variant with imperfect information and a fixed small action space. RLCard's implementation was chosen due to its open-source nature, modular desing, and widespread use.

The Leduc Hold'em environment in RLCard is a custom-built simulation designed to support research in reinforcement learning and imperfect-information games. It follows the standard 2-player, fixed-limit Leduc Hold'em poker format, with clear modular separation between game components. RLCards implementation of Leduc Hold'em is structured around several classes:

- LeducholdemGame: The main game class which allows gameplay. It manages players, dealer, rounds, and transitions between game states.
- Player: Represents each agent in the game, stores private hand information, chip count and betting status.
- Dealer: Manages card dealing, including assigning private hands and revealing the public card.
- Round: Controls the betting rounds, including turn order, allowed actions, raise limits, and round transitions.
- Judger: Determines the winner at the end of the game based on hand strength.

RLCards version of Leduc Hold'em uses the following fixed parameters described in Table 4.1:

Property	Description
Number of players	2 (fixed)
Deck	6 cards (2 of each of the 3 ranks)
Blinds	Small blind: 1 chip bet; Big blind: 2 chip bet
Raise amounts	Pre-flop: 2 chip bet; Post-flop: 4 chip bet
Raise cap	Maximum of 2 raises per player per round

Table 4.1: Leduc Hold'em Game Parameters

At the start of each game, players are randomly assigned to post the small and big blinds. A betting round is initialized, with the raise amount passed to the Round object which is responsible for managing the logic of a single betting round. After the first betting round concludes, a public card is revealed and the raise amount is doubled for the second round.

Each players state includes:

- Their private hand
- The public card (if revealed)
- Current chip counts of all players
- Legal actions available at that point
- Betting history and game pointer (i.e., whose turn it is)

This state is returned by the environment during gameplay and serves as the input to any learning agent.

4.2 RLCard Implementation of DQN

RLCard provides a modular and extensible implementation of DQN which is tailored for training agents in imperfect-information card games like poker [ZLC+19]. The PyTorch-based version closely follows the original DQN algorithm [MKS+15] with several practical adaptations to support batch training, evaluation, and integration into the RLCard framework.

At the high-level, the RLCard DQN agent consists of a Q-Network, Target Network, and an Experience Replay Buffer. Both networks are built using multilayer perceptrons with configurable depth. Particularly, this implementation of DQN uses Double DQN(DDQN). The reason for this is that one of the key limitations of the original DQN algorithm is the tendency to overestimate Q-values during training [MKS+15]. This overestimation arises because the same Q-network is used to both select the action with the highest predicted value and to evaluate that actions value. This can lead to overly optimistic estimates, especially in noisy or stochastic environments. Double DQN (DDQN), proposed by van Hasselt et al. (2016), addresses this issue by decoupling the action selection from the action evaluation [vHGS16]. In DDQN, the main Q-network is used to select the best action for the next state, but the target network is used to evaluate the value of that action.

This separation reduces bias and leads to more stable learning, especially in environments with high variance in rewards or long horizons.

The target for DDQN is computed as:

$$y = r + \gamma \cdot Q\left(s', \arg\max_{a'} Q(s', a'; \theta); \theta^{-}\right) \tag{4.1}$$

where the main network θ selects the best next action via $argmax$ while the target network θ^{-} evaluates that action.

The agent follows an ϵ-greedy policy during training, where it selects a random legal action with probability ϵ to encourage exploration, and with probability $1 - \epsilon$, it choose the action with the highest predicted Q-value. Epsilon decays linearly over a fixed number of steps which shifts the agent from being highly explorative early on in the training process to being exploitative later on.

Training is performed at regular intervals during environment interaction. At each training step, a batch of transitions is sampled, and the Q-network is updated to minimize the MSE (Mean-Squared Error) between predicted Q-values and target values. To stabilize learning, the weights of the target network are periodically updated to match those of the main Q-network.

Since the implementation is meant to be used with card games, it ensures action legality by masking out invalid actions. The predicted Q-values for illegal actions are set to $-\infty$ which ensures that they are never selected during action choice.

4.3 Implementation of CFR

To support asymmetric training scenarios and interaction with arbitrary opponent policies, we implemented a custom version of Counterfactual Regret Minimization (CFR) that allows one agent to train against a separate, externally defined opponent. This departs from the standard CFR implementation, which relies on self-play by simulating both players using the same CFR algorithm. However, it still closely follows the standard CFR implementation [ZJBP07]. In our implementation, CFR updates its strategy by traversing the game tree while treating the opponent's actions as fixed and externally provided by a callable policy interface. This enables the agent to train against any opponent, including non-tabular agents or those learning concurrently.

As in standard CFR, the agent maintains three key data structures:

- Regret Table: Stores cumulative regrets for each action at every information set.
- Current Policy: A distribution over actions, updated via regret matching based on the regret table.
- Average Policy: A running average of past policies, used for evaluation and gameplay.

During each iteration, the CFR agent performs a recursive traversal of the game tree. When it is the agent's turn to act, it selects actions according to its current policy and computes counterfactual

utilities. When it is the opponent's turn, actions are selected using the opponent's current policy via a function call. This distinction allows CFR to compute regrets relative to a fixed or dynamic opponent rather than simulating both sides itself. After traversal, regrets are updated based on the difference between actual action utilities and the expected utility, and the average policy is incrementally refined using reach probabilities.

Information sets are represented by string-encoded observation vectors, allowing generalization across structurally identical states. Legal action masking is applied to ensure only valid actions are considered in each state, which is a crucial feature in imperfect information games like Leduc Hold'em.

For evaluation and gameplay, the agent selects actions from its average policy, which converges to a more stable and robust strategy over time. The internal data structures, namely, regrets, policies, and iteration count are serialized to save at regular intervals to support checkpointing and reproducibility.

This custom CFR implementation provides a flexible framework for studying interactions with diverse opponents, making it suitable for analyzing response strategies, exploitability, and strategic adaptation in multi-agent environments.

4.4 Training of CFR and DQN on Random Agent

To assess the strategic behavior of the DQN and CFR agents, both were initially trained against a common baseline, namely, the RandomAgent from the RLCard library. This ensured a fair and consistent environment for learning. All training and evaluation were conducted using the Leduc Hold'em environment in RLCard with a fixed random seed of 42 to ensure reproducibility.

CFR Training The CFR agent was trained using the custom Counterfactual Regret Minimization implementation for 10000 iterations. Only the CFR agent was updated at this stage. The RandomAgent acted as a fixed opponent. At each iteration, the CFR agent traversed the game tree for player 0 (the CFR agent), calculated counterafactual regrets, and updated its current and average policies via regret matching. Evaluation was performed every 500 iterations over 1000 games, and win rates were logged using Weights & Biases (W&B) [Bie20]. After training, the average policy was extracted and stored in a merged file to be used for later evaluation.

DQN Training The DQN agent was trained for 10000 episodes against a fixed RandomAgent. The training used the PyTorch-based DQNAgent in RLCard, configured with a multilayer perceptron of architecture $[64, 64]$, a replay memory size of 20000, batch size of 32, learning rate of 0.00005, and a discount factor of 0.99. The agent employed a ϵ-greedy policy, where ϵ decayed linearly from 1 to 0.1. Target network updates occurred every 1000 steps and the model checkpoint was saved at the end of training.

Cross-Evaluation After training, the final models were evaluated against each other across 100000 games. DQN selected actions based on its learned Q-values, while CFR sampled from its

average policy. Evaluation results were logged at intervals of 100 games tracking win rates and draws.

4.5 Cross-training on frozen agents of DQN and CFR

The next training phase involved training DQN and CFR on frozen versions of each other. The "frozen agent" in this context refers to a agent whose policy was obtained from training against a random opponent and is then kept fixed, that is, it is not updated during training. This setup allows us to see the performance of DQN and CFR when they have some exposure to each other during training.

CFR Training Against frozen DQN The CFR agent was trained for 10000 iterations against a DQN agent that had been previously trained on Leduc Hold'em against a random opponent. The frozen DQN model was loaded from a saved PyTorch checkpoint and used as the fixed opponent during all iterations. CFR used regret matching to iteratively update its policy and regret tables. Evaluation was conducted every 500 iterations over 1000 games, logging the CFR and DQN win and draw rates to Weights & Biases (W&B). The CFR agent was allowed to use environment step-backs, and the game seed was fixed to 42 to ensure reproducibility. After training, the agents policy, average policy and regrets were stored along with a merged pickle file for evaluation use.

DQN Training Against Frozen CFR The DQN agent was trained for 10000 episodes against a frozen CFR agent, whose average policy was extracted from a previously trained model (trained against a random agent). The frozen CFR policy was loaded from a pickled dictionary of regrets and policies. The DQN agent used a multilayer perceptron with 2 hidden layers of 64 units each, and followed the standard DDQN algorithm with ϵ-greedy exploration. Epsilon decayed linearly over the training duration, starting at 1 and ending at 0.1. Evaluation was performed every 500 episodes across 1000 games, logging the win rates of both agents and the draw rates. All results were tracked using W&B.

Cross Evaluation After both agents were trained against frozen versions of each other, we conducted a direct evaluation between them. Each agent was loaded from its respective checkpoint, and 100000 evaluation games were played in the Leduc Hold'em environment. The win rates for both agents and the draw rates were recorded. This provided a performance snapshot which reflects how well each agent learned to exploit the previously trained opponent while also generalizing to a different but related adversary.

4.6 Simultaneous Training of CFR and DQN

The next training phase was done to try to ensure that both DQN and CFR would be able to have exposure of each other during training and that both would try to adapt to each other in real time. Thus, a unified training loop was implemented where both agents were trained concurrently in the same environment. This setup allowed each agent to effectively learn while interacting with a non-stationary opponent that is adapting over time.

Environment Configuration The RLCard environment for Leduc Hold'em was initialized with allow_step_back=True to enable CFR to traverse the game tree. Each training episode consists of both agents interacting in a shared environment, with CFR always assigned as Player 1 and DQN as Player 0. Prior to each gameplay episode, the CFR agent performs multiple regret update iterations by traversing the game tree while treating DQN as a fixed opponent. In contrast, the DQN agent trains post-game by storing gameplay trajectories and updating its Q-network. The agents interact in the same environment, effectively learning in tandem. All training logs, including rewards, win rates, and per-game logs from CFRs and DQNs perspective, are stored and periodically recorded to Weights & Biases.

CFR Agent Updates The CFR agent used in this setup implements custom variant that performs updates by traversing the game tree recursively. During traversal, if it is the opponent's turn (Player 0), the CFR agent queries the current policy of the DQN agent to simulate the opponent's move. If it is the CFR agent's turn, it selects actions based on regret matching and accumulates counterfactual utilities to update its regret and average policy tables. Each training episode includes a fixed number of CFR iterations before gameplay begins. The agent maintains an average strategy over time, which is used for evaluation and action selection during games. Information sets are encoded as byte strings for efficient lookup, and all action probabilities are masked to ensure legality within the game rules.

DQN Agent Updates The DQN agent uses a multi-layer perceptron with two hidden layers of 256 units each, a learning rate of 0.00005, batch size of 64, and a linear epsilon decay of 10000 steps to encourage exploration early in training. The replay buffer is initialized with 500 experiences before training begins. At each episode, the agent collects experiences by interacting with the environment and performs updates using temporal-difference learning. Evaluation is conducted every 50 episodes, during which 1000 games are played to compute win rates and reward statistics. Some hyperparameter values might differ than in previous training setups. A sweep was ran over a pre-defined set of hyperparameters and associated values, and this configuration was found to be the one that matches the performance (win rate) of CFR the closest.

Training Synchronization Each training episode consists of two phases executed sequentially. First, the CFR agent performs a fixed number of iterations using recursive tree traversal, updating its regrets and average policy while treating the opponent's actions as determined by the current DQN policy. Following this, a complete game is played between the two agents. The DQN agent then updates its Q-network based on the resulting trajectories and rewards collected during gameplay. This setup creates a feedback loop in which both agents continuously adapt to one another's evolving strategies within the same training cycle.

Evaluation Finally, after 10000 training episodes, the trained models are saved, and an evaluation phase is conducted where 100000 games are played with the agents using their learned (average) policies. These games are logged from both agents' perspectives for further analysis of bluffing behavior. This simultaneous training experiment allows us to investigate not only learning performance, but also how different agent architectures adapt to the strategies of non-stationary opponents over time.

4.7 Bluff Detection via Heuristic Criteria

To assess whether bluffing occurred, we define a simple but effective heuristic that classifies certain raise actions as bluffs based on private hand strength and public card information. The core idea is that a bluff occurs when an agent aggressively raises despite holding a weak hand. It is an action that would not be justified based on hand strength alone.

Each agents private hand is evaluated using a hand_strength() function which assigns a base score to the card rank. Jack (J) is 1 point, Queen (Q) is 2 points, and King (K) is 3 points. If the rank matches the public card, and thus forms a pair, the score is boosted by 3 points which reflects the increased value of having a pair in poker.

The is_bluff function then determines whether a particular action qualifies as a bluff. Specifically, if an agent performs a raise action while holding a hand with a strength score of 2 or lower (non-paired J or Q), the action is classified as a bluff.

This rule-based detection is used to flag and count bluff attempts during evaluation games, which enables comparative analysis of bluff frequency, bluff success, and opponent reaction.

4.8 Summary of Methodology

In this methodology section we have looked at the environment configuration, the implementation of both the DQN and CFR agents, and the three training setup that we used. Lastly, we looked at the bluffing heuristic that we defined in order to filter bluffs which will make the bluffing analysis possible. In the next chapter we will provide the results.

Chapter 5

Experiments

In order to evaluate the bluffing tendencies of the Deep Q-Network (DQN) and Counterfactual Regret Minimization (CFR) agent in Leduc Hold'em, we analyzed bluff frequency, bluff success rates, card-specific bluffing patterns, and timing of the bluff relative to the reveal of the public card. The evaluation was conducted with bluff detection based on the heuristic described in section 4.7, and success determined by the opponent folds.

5.1 Training

Initially, both CFR and DQN were trained against a random agent. Then the trained agents would be loaded and evaluated against each other in a series of Leduc Hold'em games. Both agents were trained on 10000 episodes of Leduc Hold'em. One episode represents 1 game of Leduc Hold'em that is played out until there is a winner. Furthermore, to calculate the win rates for the games against the random agent, a 1000 games of Leduc Hold'em is played in evaluation mode (is_training is set to False) after every 500 episodes (games) of training.

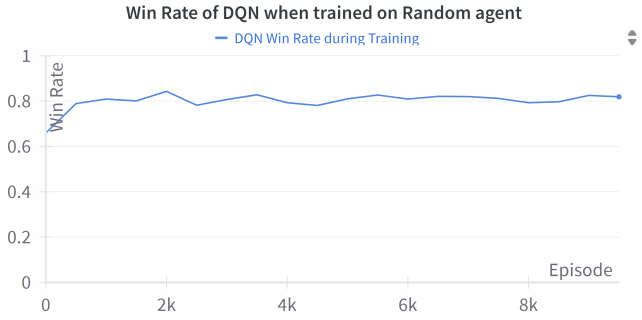


Figure 5.1: Win rate of the DQN agent in Leduc Hold'em when trained against a Random agent. The plot shows that the DQN agent quickly learns to outperform the Random opponent, achieving a win rate above 80% after only a few thousand episodes and maintaining strong performance with minor fluctuations throughout training.

From Figure 5.1, we can see how the DQN agent stabilizes around 80% win rate against the random agent. DQN is beating the random agent consistently. However, the win rate is not 100%

as DQN can never truly learn how to fully exploit the random agent as the random agent has no pattern of moves nor strategy, it is purely random. Additionally, sometimes hands are dealt that are unwinnable.

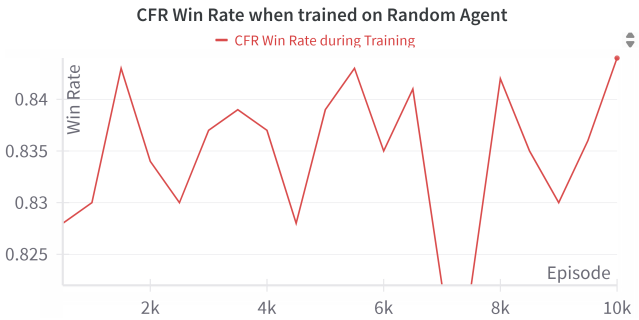
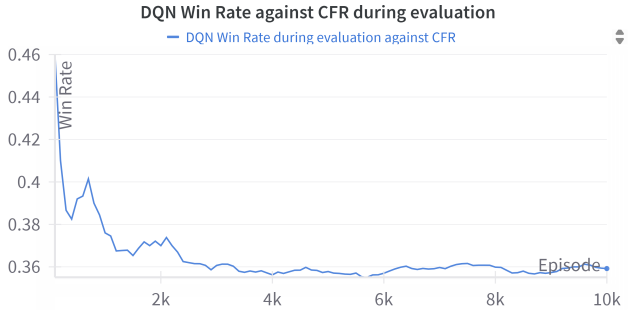
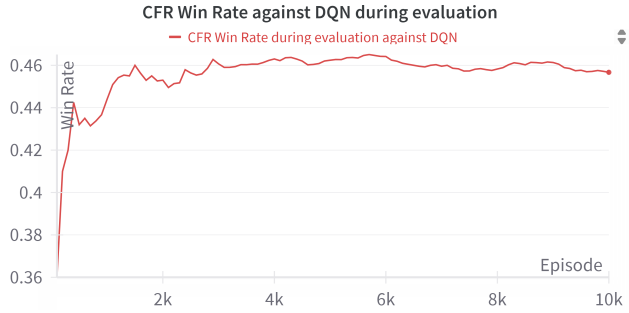


Figure 5.2: Win rate of the CFR agent in Leduc Hold'em when trained against a Random agent. The CFR agent shows exponential improvement in win rate during training, reaching approximately 83% by 2000 episodes.

From Figure 5.2 we can see that the win rate of CFR stabilizes roughly between 82% and 84% against the random agent. This implies that CFR is beating the random agent most of the time, however, not all of the time. Firstly, the random agent has no pattern nor tendency to play certain moves and thus, it cannot be exploited to a significant degree. The CFR agent cannot be exploited anymore but neither can the CFR agent exploit the random agent further.



(a) DQN win rate against CFR.



(b) CFR win rate against DQN.

Figure 5.3: Win rates of DQN and CFR agents when evaluated against each other after training exclusively on a Random agent. The DQN agent achieved a win rate of $\approx$ 36%, the CFR agent $\approx$46%, and the remaining $\approx$18% of games ended in draws.

After both agents have been fully trained on the random agent, the DQN and CFR agents played against eachother in 100 000 games of Leduc Hold'em. From this, the DQN agent had a win rate of approximately 36%, while the CFR agent had a win rate of approximately 49%. The draw rate was around 18%. Although, this training setup provided a simple and consistent training environment, it was inherently unbalanced which can be seen from the final evaluation win rates. Neither agent has been exposed to strategic behavior during training, nor did they have the opportunity to adapt

to each others play style. To address this limitation, we introduced a second version of training where both agents were trained on a frozen version of the other, and then evaluated in the same manner. Frozen here means that the agents policies do not change during training. The policy is fixed/static and the frozen agent does not learn or adapt while the other agent is being trained and its policy is being modified.

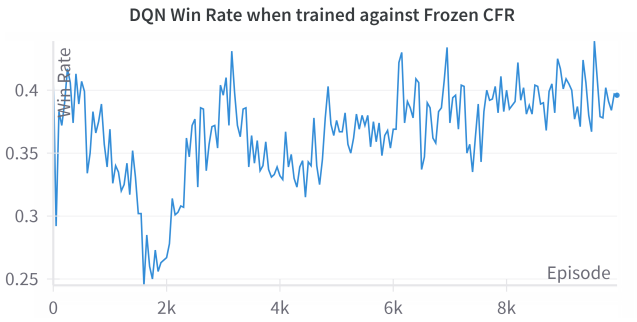


Figure 5.4: Win rate of the DQN agent in Leduc Hold'em when trained against a frozen CFR opponent. The DQN agent at the start achieves a win rate slightly above 40%, but its performance declines and remains unstable throughout training stabilizing in the range between 35% and 42%.

From Figure 5.4 we can see the win rate of the DQN agent when it is trained against a frozen CFR agent policy through 10 000 episodes of Leduc Hold'em. We observe significant volatility in win rates and an early decline in performance. This behavior is expected due to the nature of the DQN algorithm. The agent updates its Q-network using stochastic mini-batches from replay memory, which can cause unstable policy shifts and performance swings. Additionally, early performance may appear deceptively strong due to random exploratory behavior that occasionally outperforms a conservative CFR policy. As training progresses, DQN starts to form a more consistent policy, but this learning process introduces variance which leads to fluctuations in win rate.

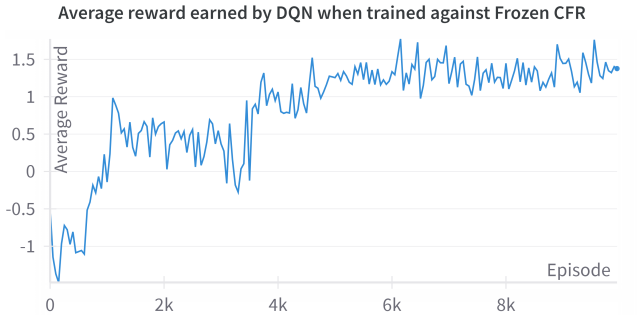
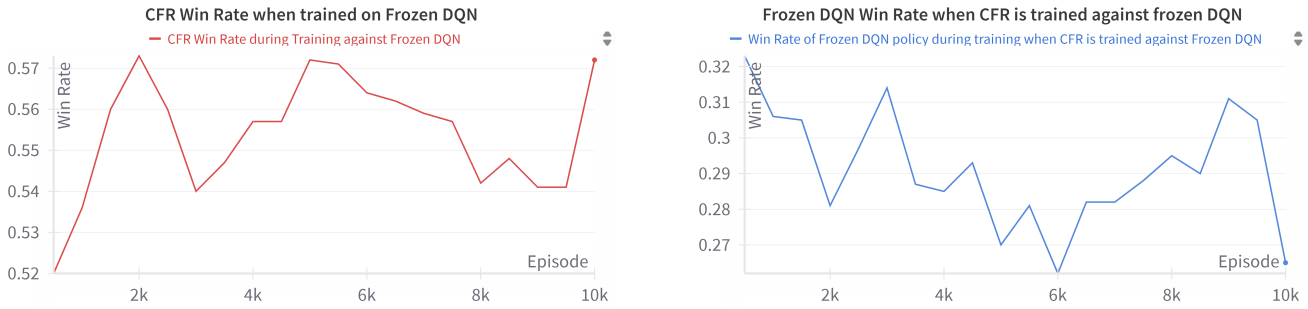


Figure 5.5: Average reward earned by the DQN agent during training against a frozen CFR opponent in Leduc Hold'em. Despite significant noise, the trend shows gradual improvement from an initial negative reward (–1) towards positive rewards, suggesting the DQN agent is slowly learning to minimize its losses against the stronger CFR policy.

Furthermore, when we consider the DQN agents average evaluation reward over time, a clearer pattern of learning emerges. As shown in Figure 5.5, the average reward gradually increases,

transitioning from negative to positive values. This trend suggests that, despite noisy win rate curves, the DQN agent is in fact learning to extract more value per game against the fixed CFR strategy.



(a) CFR win rate vs. frozen DQN

(b) Frozen DQN win rate vs. CFR

Figure 5.6: Win rates during training of CFR against a frozen DQN agent in Leduc Hold'em. (a) CFR maintains its win rate over time, after learning quickly at the start. (b) In parallel, the performance of the frozen DQN agent declines as CFR learns to exploit its static strategy.

From Figures 5.6a and 5.6b we observe that CFR can improve significantly over time when trained on a frozen DQN policy. The frozen DQN policy that we are using here is the policy that DQN has learned from training against the random agent. We observe that DQNs win rate drops from 32% to 27%, while CFRs win rate increases from 52% to approximately 57% which indicates learning and adaptation.

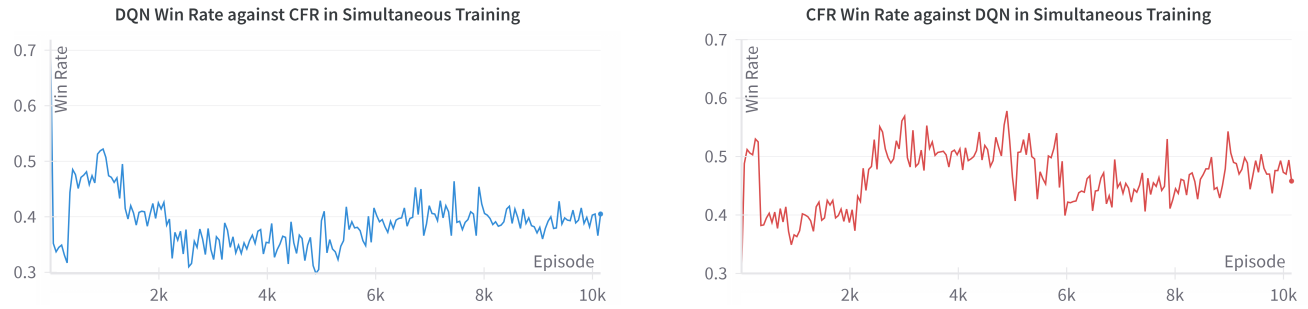
Despite the static nature of the DQN opponent, the CFR agent is able to gradually increase its win rate over time, eventually surpassing the frozen DQN's performance. This aligns with CFR's theoretical foundation as a no-regret learning algorithm that converges to a Nash equilibrium in two-player zero-sum games. Although the DQN policy was trained against a random opponent and may contain suboptimal or exploitable behaviors, CFR is shown to adapt effectively by minimizing its regret over time and adjusting to the DQN's fixed weaknesses. Interestingly, while CFR does not explicitly optimize for win rate (unlike DQN), it still achieves superior performance by converging to a strategy that balances robustness with effective responses to DQN's play. The rise in CFR's win rate, from roughly 52% to 57%, and the corresponding drop in DQN's win rate, from 32% to 27%, demonstrate this adaptive advantage.

However, these results also show how the CFR agent is unable to completely dominate the frozen DQN agent, which is seen by the CFR agent having a 57% win rate, and the draw rate being around 16%. This is because CFR does not learn to exploit agents, but only to play a minimally regretful response to it, as it is a game theory based algorithm. Since the frozen DQN policy (trained on a Random agent) has no guarantees of reaching the Nash equilibrium it may contain aggressively exploitative patterns and thus the CFR agent might converge towards a defensive, regret-minimizing strategy that neutralizes DQN in some situations but fails to punish its weaknesses effectively. CFRs objective is to minimize cumulative regret which is in contrast to the RL paradigm which directly optimizes for win rate. This means that CFR aims for robustness and balance and not

exploitation. So even with full visibility of DQNs behavior, CFRs learning dynamics possibly lead to a conservative policy that defends and wins but does not dominate.

For the bluff analysis we want the agents to directly adapt to each other in real time, instead of using proxies of each other. Thus, we did a third training phase where we let both agents train against each other simultaneously. This setup allows for mutual adaptation, where each agent learns not only to defend against the others strategy but also to refine its own behavior in response to strategic shifts over time. By co-evolving their strategies during training, we encourage the emergence of more nuanced and competitive dynamics which is critical for capturing meaningful bluffing behavior and for evaluating how each agent reacts to deceptive play.

In the final training phase, both the CFR and DQN agents were trained simultaneously against each other with the aim of allowing both agents to adapt to eachother. Furthermore, it was also done to create a more balanced competitive environment which would allow for more natural and interpretable bluffing behavior to emerge rather than strategies being skewed by one-sided exploitation.



(a) DQN win rate vs. CFR

(b) CFR win rate vs. DQN

Figure 5.7: Win rate dynamics during simultaneous training of DQN and CFR agents in Leduc Hold'em. (a) DQN initially gains an edge, but its win rate declines as CFR adapts. (b) CFR steadily improves, reaching a win rate of 40–50%.

From Figure 5.7a we can see that DQN achieves a win rate of approximately 70%, indicating an initial advantage, likely due to faster convergence or stronger early exploitation. However, as training progresses, CFR begins to adapt and DQNs win rate declines to around 35%-45%. Simultaneously, CFR's win rate rises and stabilizes in a band between 40% to 50%, and briefly peaks to 55%, which can be seen from Figure 5.7b. From around 6000 episodes onward, the DQN agent oscillates around 40% win rate, while the CFR agents win rate stays between 40% and 50%. This behavior indicates mutual adaptation and convergence to a near-equilibrium state, which was not observed in the earlier training phases. The ability of both agents to adjust in real time results in a dynamic learning environment where neither agent completely dominates the other.

This final training phase demonstrates that CFR and DQN can indeed be successfully cross-trained in a simultaneous setting, allowing both agents to adapt to each other in real time. Unlike the earlier frozen-agent setups simultaneous training facilitates mutual adaptation. This setup enables a fair

and symmetric evaluation of bluffing behavior which will be discussed in the following subchapter.

5.2 Do CFR and DQN bluff?

With the agents successfully co-trained in a balanced and competitive environment, we turn our attention to one of the most strategically rich aspects of poker, bluffing. Bluffing is not only a fundamental part of poker gameplay, but also a revealing indicator of how well agents understand risk, deception and opponent modeling. In this section, we analyze whether bluffing emerges naturally in the learned strategies of both the DQN and CFR agents.

After the simultaneous training phase, the agents played 100 000 games of Leduc Hold'em against each other in evaluation mode. All of those games were logged and the following resulted.

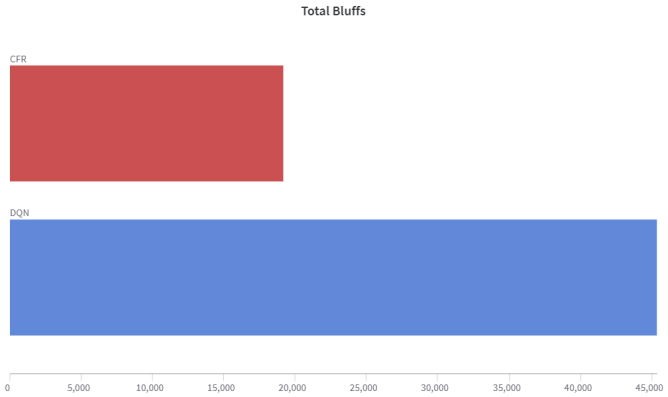


Figure 5.8: Total number of bluffing actions observed during evaluation. Each bar represents the number of bluffs performed by one agent across 100,000 Leduc Hold'em games after simultaneous training. The DQN agent (blue) bluffs significantly more than the CFR agent (red), indicating that bluffing has become a more prominent and consistent tactic in the DQN's learned strategy.

From Figure 5.8 we observe a clear asymmetry in bluffing behavior, the DQN agent bluffs substantially more often than the CFR agent. This suggests that bluffing is not just incidental but a core component of the DQN agent's learned policy. Since DQN optimizes directly for reward, it may discover that aggressive bluffing yields favorable outcomes against certain opponent behaviors. In contrast, the CFR agent, which minimizes regret and seeks robustness rather than exploitation, appears more conservative in its bluffing frequency. These results highlight a key difference in strategic tendencies between reinforcement learning and game-theoretic approaches under simultaneous training.

To better understand the bluffing tendencies of each agent, we analyzed the total number of bluffs that have been made according to the private hand held during the bluffing action.

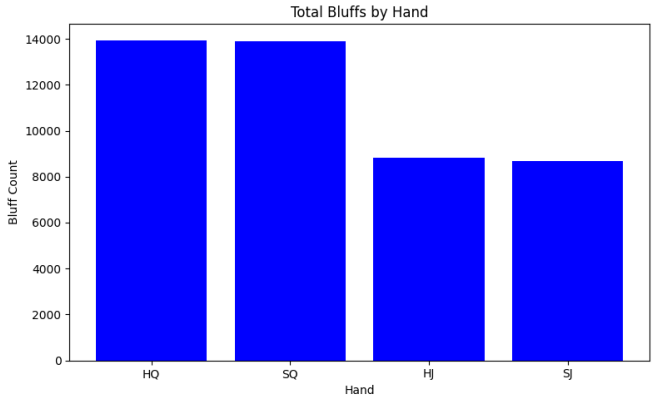


Figure 5.9: Distribution of bluffing actions made by the DQN agent, grouped by private hand. Bluffing is more frequent with Queen hands (HQ and SQ), but Jack hands (HJ and SJ) are also used to bluff in a substantial number of cases. This indicates that the DQN agent employs a relatively broad, yet still selective, bluffing range, favoring hands that lie near the boundary of value and bluff.

From Figure 5.9 we observe that the DQN agent shows a clear preference for bluffing with Queen hands (HQ and SQ), with each hand contributing over 13,000 bluffing instances. However, the agent also bluffs with Jack hands (HJ and SJ) in nearly 9,000 instances each, suggesting that DQN maintains a fairly diverse bluffing strategy. Rather than avoiding weak hands entirely, the DQN agent appears to consider both Queens and Jacks as viable bluffing options, potentially balancing the perceived risk of being called with the reward of winning uncontested pots. This behavior implies that DQN has learned to bluff with a range of hands that are not strong enough to value bet but not so weak that the bluff is easily punished. The inclusion of Jack hands in the bluffing repertoire points to a more nuanced strategic balance between exploitation and risk management.

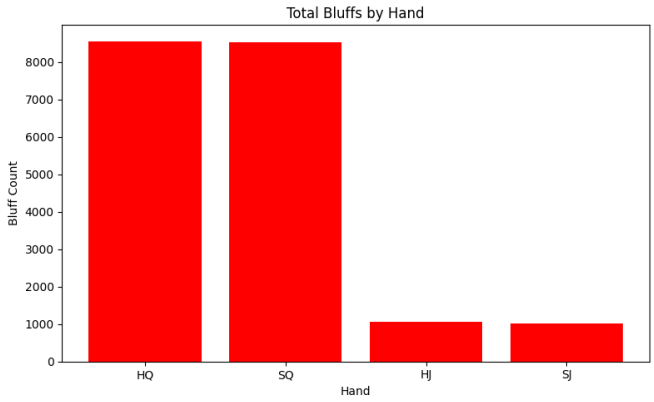


Figure 5.10: Distribution of bluffing actions made by the CFR agent, grouped by private hand. Bluffing is heavily concentrated around Queen hands (HQ and SQ), while Jack hands (HJ and SJ) are rarely used for bluffs. This indicates that the CFR agent has adopted a narrow bluffing range, likely favoring hands that are weak but still have some showdown value.

Figure 5.10 shows that Queen hands (labeled as HQ and SQ) account for the vast majority of

bluffs made by the CFR agent. In contrast, Jack hands (SJ and HJ) are used far less frequently for bluffing. This suggests that CFR adopts a relatively narrow bluffing range, preferring to bluff with slightly stronger weak hands rather than employing a broad distribution. Such behavior may reflect an effort to balance risk and predictability, bluffing with Queens offers some residual value if called, while rarely bluffing with Jacks may avoid overly weak or easily punished bluffs. The agents avoidance of J hand bluffs may indicate that internally the agent has learned the risk to be too high compared to the chance of a bluff succeeding and therefore it has learned J hands as not worth the strategic cost. This pattern highlights CFR's inclination toward cautious, regret-minimizing play rather than aggressive deception.

While the frequency and distribution of bluffing actions offer valuable insight into each agents bluffing tendencies, bluffing in isolation does not capture the full picture. A successful bluff is one which elicits the desired reaction from the opponent, typically a fold. Therefore, a critical component of understanding bluffing behavior comes from analyzing how their opponents respond to the attempted bluffs. In the next section we will examine the opponents responses to bluffs.

5.3 What are the agents reactions to bluffing?

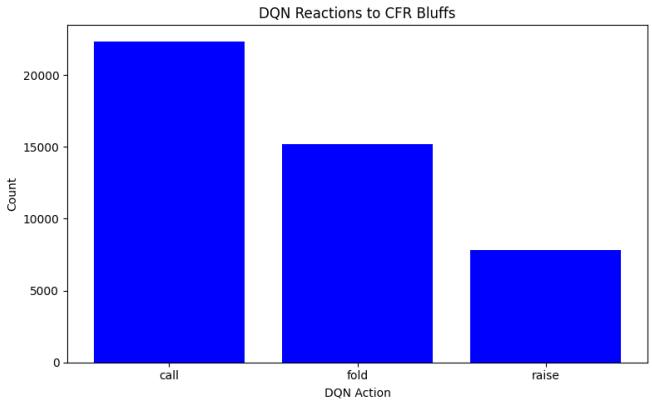
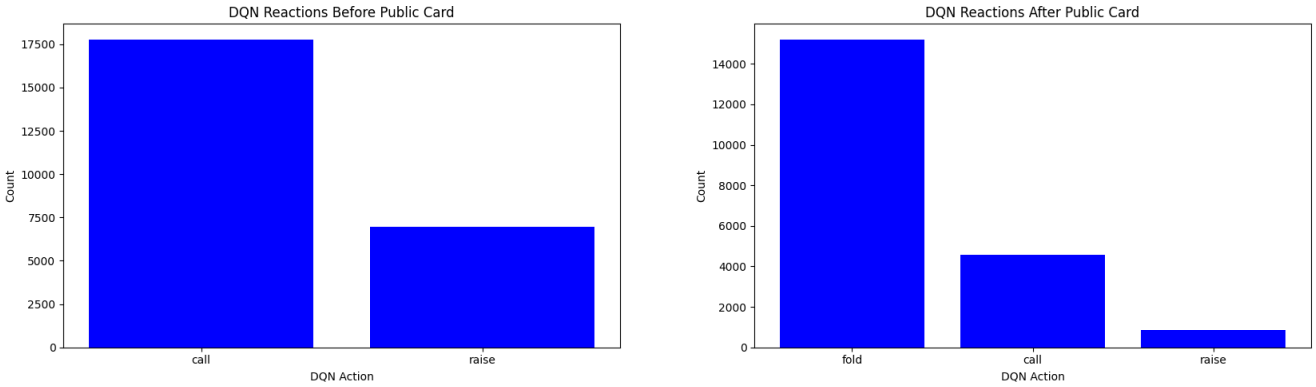


Figure 5.11: Distribution of DQN agent's reactions to bluffing attempts by CFR. The majority of responses are calls, indicating a passive but curious strategy, followed by a moderate number of folds and fewer raises.

Figure 5.11 gives insights into the responses of DQN when CFR attempts to bluff. The most common reaction by far is calling with over 20 000 instances. This indicates that DQN frequently chooses to see through the bluff rather than retreat or escalate. This behavior suggests that DQN, as a reinforcement learning based agent, may have learned to treat uncertain situations more conservatively by defaulting to passive observation. However, this tendency to call may also reflect an incomplete or overly cautious bluff detection strategy of the agent, where the agent avoids folding too frequently but also fails to assertively punish bluffs with raises. Interestingly, folds occur in fewer than 15 000 cases which means that CFRs bluffs succeed in making DQN fold less than half the time. From CFRs perspective (the bluffer), this implies a moderate bluff success rate against DQN. The relatively low number of raises may indicate that DQNs strategy is focused on information

gathering and long-term stability rather than aggressive counterplay. All of this shows that the DQN agent displays a balanced but cautious attitude toward opponent bluffs, preferring to call and observe what happens rather than to commit too heavily to either folding or counterattacking. This strategy may minimize risk over time but could also fail to capitalize on opportunities to punish systematic bluffing.

In order to gain a more granular understanding of how the DQN agent responds to bluffs, we analyzed its behavior across the 2 different phases of the game, namely, before and after the public card is revealed. The results show a stark shift in response strategy.



(a) Before public card

(b) After public card

Figure 5.12: DQN agent reactions to CFR bluffing across game phases. (a) Exploratory behavior with calls and raises dominates before the public card. (b) Defensive folding behavior dominates after the public card is revealed.

From Figure 5.12a we can observe that in the pre-public card phase (the first betting round), DQN demonstrates a relatively balanced and active stance. It calls nearly 18000 times and raises around 6500 times with no recorded folds. This behavior highlights that in the early-stage betting, where hand information is limited and uncertainty is high, DQN prefers to stay in the game and collect additional information, rather than fold its hand prematurely. Its willingness to raise at this stage also shows that DQN is not passive and occasionally opts to apply pressure, possibly to probe for information or deter opponent aggression. The situation shifts dramatically after the public card is revealed which can be seen from Figure 5.12b. In this phase, folding becomes the dominant reaction, with over 15 000 instances, while calls and raises drop sharply to only a few thousand each. This sharp increase in folding frequency suggests that DQN adopts a far more cautious and risk-averse strategy once additional board information is available and the stakes are higher.

The contrast between these 2 graphs indicates that DQN is phase-sensitive in its bluff response strategy. It is relatively exploratory and aggressive before the public card, but becomes defensive and conservative after additional information is revealed.

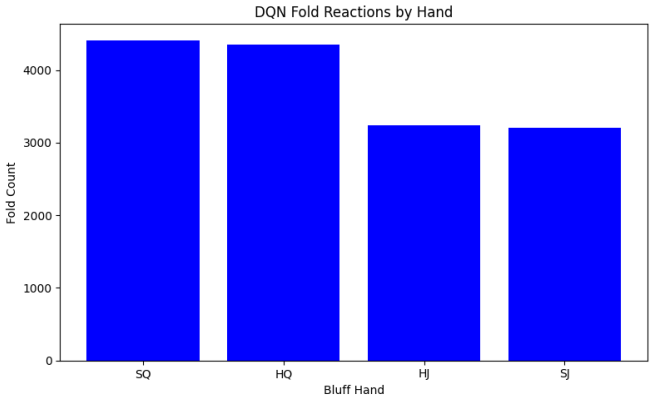


Figure 5.13: Number of DQN folds in response to CFR bluffing, categorized by the bluffing hand. The agent folds most often against Queen hands, indicating greater respect for mid-strength bluffs, while folding less frequently to Jack hands.

From Figure 5.13 we can see how often the DQN agent folds in response to CFR bluffs, broken down by the hand used in the bluff. The results reveal that folds are the most common against Q hands, each causing over 4000 hands, while J hands trigger fewer folds. This suggests that DQN is more likely to respect bluffs from mid-strength hands, likely perceiving them as more credible or risky. In contrast, bluffs with weaker hands are more often challenged, indicating that DQN has possibly learned to somewhat distinguish between believable and unconvincing bluffs.

We will now analyze the reactions of the CFR agent to the bluffing attempts of the DQN agent.

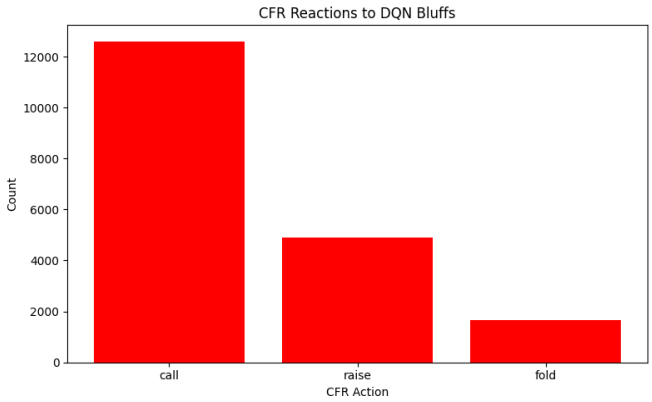
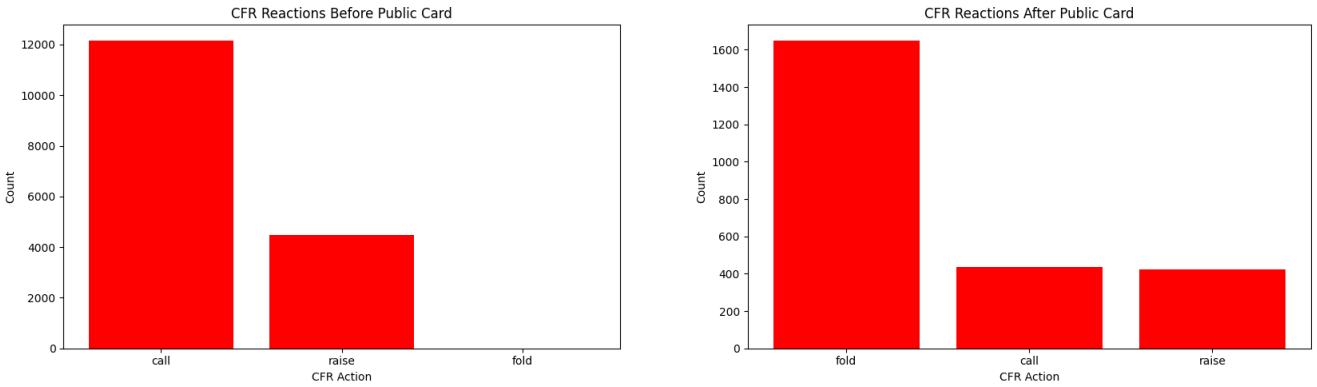


Figure 5.14: Distribution of CFR agent's reactions to bluffing attempts by DQN. The agent responds mostly with calls, followed by raises, and only rarely folds, reflecting a confident and resilient defensive policy.

Figure 5.14 shows that the CFR agent most frequently respond to DQN bluffs by calling with over 12000 calls recorded. This suggests that CFR prefers to verify the opponents bet rather than to fold outright, potentially minimizing regret by avoiding premature concessions. Raises are the second most common response, occurring over 5000 times which indicates that CFR is also capable of applying pressure when suspicious of a bluff. Folds are the least frequent reaction, with fewer than

2000 occurrences which implies that DQNs bluffs are not highly effective at forcing CFR to back down. Overall, CFR exhibits a confident and resilient defensive strategy.

To get a more nuanced understanding of bluff dynamics, we will analyze the reactions of CFR to DQN bluffing per phase of the game.



(a) Before public card

(b) After public card

Figure 5.15: CFR reactions to DQN bluffing by phase. The agent is aggressive before the public card (left), favoring calls and raises. After the public card (right), CFR becomes more conservative, introducing folds and reducing aggression.

From Figure 5.15a we can see that before the public card, CFR overwhelmingly favors calling, with over 12000 calls and raises nearly 5000 times. There are no folds recorded in this phase which indicates that CFR prefers to stay in the game during early play to gather more information and minimize regret from folding prematurely. From Figure 5.15b, we observe that after the public card is revealed CFRs behavior shifts noticeably. Folding becomes the dominant action, while calling and raising both occur less frequently. This indicates that CFR becomes more conservative once more information is available, choosing to fold in unfavorable situations or when DQNs betting appears strong.

From this per game phase CFR reaction to DQN bluffing analysis we can see that CFR plays more defensively post-flop, incorporating folds where appropriate, but maintains a strong presence in the game pre-flop through aggressive calling and raising.

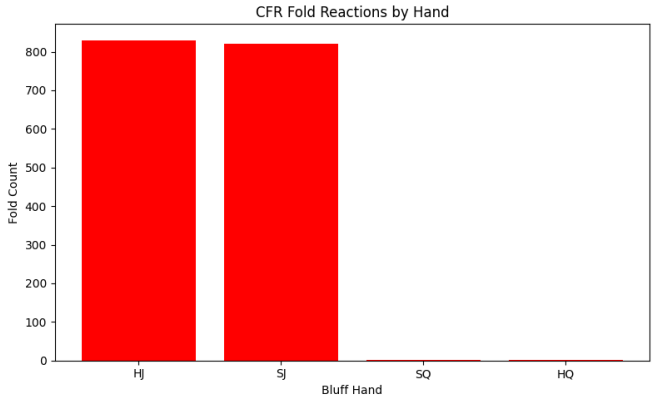


Figure 5.16: Number of folds by CFR agent in response to DQN bluffing, broken down by the bluff hand. Folds mostly occur against Jack hands (HJ and SJ), indicating that CFR challenges most Queen-hand bluffs and folds selectively to weaker ones. Folds to Queen hands are neligible.

Figure 5.16 shows that the CFR agent folds almost exclusively to bluffs made with Jack hands. Notably, the very small number of folds recorded in response to Queen hands suggests that CFR is more likely to challenge stronger-looking bluffs and only folds when the bluff is constructed from hands that represent the lowest possible strength. This indicates that CFR uses a risk-aware strategy, folding selectively and only when the betting behavior aligns with a weak hand that does not credibly represent strength. In effect, CFR shows a pattern of skepticism towards high-card bluffs and a willingness to fold only when the hand-value combined with the bluffing action fails to justify continued play.

The analysis of both DQNs and CFRs reactions to bluffs reveals important differences in how the two agents handle deception. DQN tends to react more cautiously post-flop, folding significantly more often when the public card is revealed, while being much more resistant pre-flop. Its folding strategy also varies clearly with bluff hand strength, showing a well-tuned sensitivity to opponent behavior. On the other hand, CFR adopt a more assertive and consistent defensive style, rarely folding overall and doing only when facing extremely weak bluffing hands such as Jack hands. This contrast underscores the strategic diversity between two agents who are based on different paradigms, where DQN adapts probabilistically through experience, and CFR leans on systematic regret minimization. Together, these findings highlight how each agent balances risk and information when responding to bluffs, shaping their decision-making across different stages of the game.

5.4 Summary of Experiments

Our experiments reveal distinct bluffing patterns between CFR and DQN agents. Firstly, we have gone over our training setup and the different training phases that we have developed. Then we reveled the bluffing occurrences that had happened in evaluation from both the DQN and CFR agents when they were trained simultaneously on each other. Lastly, we looked at the reactions that each of these agents had when the other agent attempted a bluff. To conclude, DQN plays a lot more aggressive, while CFR consistently plays more conservative.

Chapter 6

Conclusions and Further Research

This thesis sets out to explore bluffing behavior in Leduc Hold'em by training and evaluating two fundamentally different agents, namely, Deep Q-Networks (DQN) and Counterfactual Regret Minimization (CFR). Through a structured series of experiments, including training on random opponents, frozen policies, and ultimately against each other simultaneously, we investigated not only whether these agents are capable of bluffing, but also how they respond to being bluffed. In this chapter, we summarize the key findings, discuss their implications, and reflect on the broader significance of algorithmic bluffing in strategic decision-making.

6.1 Summary and answers to RQ

The first research question was: "How should CFR & DQN agents be trained?"

The CFR and DQN agents should be cross-trained on each other to ensure that they have complete exposure to each other. The results showed that both agents are capable of adapting to fixed versions of their opponent (frozen agents), and more notably, they can be trained simultaneously without either agent collapsing or dominating consistently. In simultaneous training, win rates between the agents stabilized within a competitive range, generally fluctuating around 50%, with neither agent consistently outperforming the other. This balance indicates that both algorithms are able to co-evolve in a dynamic environment, learning from each others evolving strategies in a stable manner. Lastly, the simultaneous training setup allows for more natural and interpretable bluffing behavior to emerge rather than strategies potentially being skewed by one-sided exploitation.

The second research questions was: "Do CFR and DQN bluff?"

Both CFR and DQN exhibit bluffing behavior. Bluffing was detected by using our bluffing heuristic. The data confirmed that bluffs occurred frequently across both agents, and that they were not random. DQN tended to bluff most aggressively with Queen hands, while CFR bluffed more evenly across weaker hand types, showing more strategic restraint. Importantly, bluff frequency remained meaningful and persistent.

The third research question was: "What are the agents reactions to bluffing?"

The reactions to bluffing varied between agents and were highly dependent on game phase and bluff strength. DQN showed a high tendency to fold after the public card, particularly when facing Queen hands. In contrast, CFR rarely folded and typically responded to bluffs with calls or counter-raises which reflects its inherently conservative regret-minimizing strategy. Interestingly, CFR only folded when facing bluffs made with the weakest possible hand, suggesting a threshold-based reaction. These findings highlight that both agents are not only capable of bluffing but also demonstrate consistent and distinguishable behavioral patterns in how they defend against deception.

6.2 Limitations

While this thesis provides meaningful insights into bluffing behavior in reinforcement learning agents, several limitations should be acknowledged.

Firstly, all experiments were conducted in the Leduc Hold'em environment which, while well suited for research due to its simplicity and interpretability, is still a vastly reduced version of real-world poker. The limited action space, card combinations and betting rounds constrain the complexity of strategic behavior and bluffing patterns.

Secondly, due to practical resource limitations, especially in regard to training time and computational power, only a finite number of hyperparameter configurations and training episodes could be explored. It is possible that longer training or alternative configurations could yield stronger or more refined behaviors.

Thirdly, the DQN implementation used a standard fully connected feedforward neural network. More advanced architectures could have allowed better information tracking and sequential reasoning. Similarly, the CFR agent was based on tabular updates rather than function approximation which limited its ability to generalize across similar states.

Fourth, the bluff detection in this thesis relied on predefined heuristics that were based on a scoring system. While effective, these heuristics may not capture subtler or more context-dependent forms of bluffing. Moreover, the heuristic might have captured some noise and treated it as bluffs.

Lastly, the thesis did not compare agent behaviors to those of human players. While this was beyond the scope of this project, such a comparison could provide a valuable frame of reference for evaluating the realism and interpretability of bluffing strategies learned by the agents.

Acknowledging these limitations is important in contextualizing the scope and generalization of the findings. Nonetheless, the controlled setup allowed for a focused and interpretable investigation into the emergence and analysis of bluffing in multi-agent reinforcement learning.

6.3 Future work

This thesis has laid the groundwork for analyzing bluffing behavior in algorithmic agents through the lens of Leduc Hold'em, but it also opens up several possible extensions for future research.

Namely, one natural extension is to apply the same bluff detection and response methodology to more complex variants of poker, such as Limit Texas Hold'em or No-Limit Texas Hold'em. These games introduce additional rounds of betting, a wider range of hands, and richer bluffing opportunities, which provide a more realistic and challenging testbed for strategic learning and deception.

Furthermore, in this thesis CFR was implemented with a tabular policy representation. Future work could explore Deep CFR or other function approximation techniques to enable generalization across similar states and hands. This could also allow CFR to scale to larger game environments and become more adaptable during simultaneous training.

Moreover, the bluff detection heuristic used in this study was rule-based and relatively rigid. A valuable direction would be to develop learned bluff classifiers using supervised or unsupervised learning on trajectory data. Such classifiers could potentially identify subtle or multi-step bluffing patterns that are not captured by our scoring system alone.

Lastly, a particular exciting direction is to test these bluffing agents against human players. This would not only provide a benchmark for evaluating strategic sophistication but also allow researchers to study how humans interpret and respond to algorithmic bluffing, potentially revealing gaps or biases in human understanding of machine behavior.

Exploring these directions would enhance the understanding of bluffing in AI systems, contribute to the development of more robust agents in imperfect-information settings, and bridge the gap between human and algorithmic strategic behavior.

Bibliography

[BBLG20] Noam Brown, Anton Bakhtin, Adam Lerer, and Qucheng Gong. Combining deep reinforcement learning and search for imperfect-information games. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[Bie20] Lukas Biewald. Weights and biases. https://wandb.ai, 2020. Software available from wandb.ai.

[BS19] Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. *Science*, 365(6456):885–890, 2019.

[CA06] Bill Chen and Jerrod Ankenman. *The Mathematics of Poker*. ConJelCo LLC, Pittsburgh, PA, 2006.

[GYY+23] Jiaxian Guo, Bo Yang, Paul Yoo, Bill Yuchen Lin, Yusuke Iwasawa, and Yutaka Matsuo. Suspicion-agent: Playing imperfect information games with theory of mind aware gpt-4. *arXiv preprint arXiv:2309.17277*, 2023.

[HMVH+18] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad G Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[MSB+17] Martin Moravčík, Matej Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2nd edition, 2018.

[SBL+05] Finnegan Southey, Michael P. Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, and Chris Rayner. Bayes' bluff: Opponent modelling in poker. In

Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI), pages 550–558, 2005.

[Skl87] David Sklansky. *The Theory of Poker*. Two Plus Two Publishing, Las Vegas, NV, 1987.

[VBC+19] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[vHGS16] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30. AAAI, 2016.

[vNM44] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, NJ, 1944.

[WD92] Christopher JCH Watkins and Peter Dayan. Q-learning. In *Machine Learning*, volume 8, pages 279–292. Springer, 1992.

[ZJBP07] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Proceedings of the 21st International Conference on Neural Information Processing Systems*, NIPS'07, page 1729–1736, Red Hook, NY, USA, 2007. Curran Associates Inc.

[ZLC+19] Daochen Zha, Kwei-Herng Lai, Yuanpu Cao, Songyi Huang, Ruzhe Wei, Junyu Guo, and Xia Hu. Rlcard: A toolkit for reinforcement learning in card games. *CoRR*, abs/1910.04376, 2019.