Analysis of Bluffing by DQN and CFR in Leduc Hold'em Poker

Tarik Začiragić, Aske Plaat, and K. Joost Batenburg

LIACS, Leiden University, The Netherlands
tarikzacir@gmail.com

Abstract. In the game of poker, being unpredictable, or bluffing, is an essential skill. When humans play poker, they bluff. However, most works on computer-poker focus on performance metrics such as win rates, while bluffing is overlooked. In this paper we study whether two popular algorithms, DQN (based on reinforcement learning) and CFR (based on game theory), exhibit bluffing behavior in Leduc Hold'em, a simplified version of poker. We designed an experiment where we let the DQN and CFR agent play against each other while we log their actions. We find that both DQN and CFR exhibit bluffing behavior, but they do so in different ways. Although both attempt to perform bluffs at different rates, the percentage of successful bluffs (where the opponent folds) is roughly the same. This suggests that bluffing is an essential aspect of the game, not of the algorithm. Future work should look at different bluffing styles and at the full game of poker. Code at `https://github.com/TarikZ03/Bluffing-by-DQN-and-CFR-in-Leduc-Hold-em-Poker-Codebase`.

Keywords: Poker · Leduc Hold'em · Bluffing · Reinforcement Learning · Game Theory · Counterfactual Regret Minimization (CFR) · Deep Q-Networks (DQN) · Imperfect-information games

1 Introduction

Bluffing is a key feature of imperfect-information games, where players must act unpredictably under uncertainty about opponents resources, intentions, or private cards. In poker, decisions depend not only on actual hand strength but also on beliefs about how opponents perceive and react. Successfully executing a bluff involves both masking weakness and exploiting the opponents uncertainty. These are skills that are traditionally associated with human intuition and behavior [5]. However, with recent advances in artificial intelligence and reinforcement learning, the question arises: Can algorithmic agents also learn to bluff?

This paper investigates the emergence and characteristics of bluffing behavior in two widely studied AI algorithms, namely, Deep Q-Networks (DQN) [3] and Counterfactual Regret Minimization (CFR) [17]. The two algorithms are based on opposite principles: reinforcement learning is reactive, training its policy on feedback, where game theoretic algorithms are based on (forward looking) first

principles analysis of the game. They were chosen because they represent the foundational methods in their respective domains. In order to facilitate extensive experimentation, we use the simplified poker environment of Leduc Hold'em, a version that includes essential poker elements such as hidden information, betting, and deception. Agents were trained against each other simultaneously, allowing mutual adaptation, and later they were evaluated against each other. The information that was logged from the games were then used for bluffing analysis.

By studying bluffing tendencies and reactions of DQN and CFR, this paper sheds light on how artificial agents handle uncertainty and deception.

The key contributions of this work are as follows:

- Using a Threshold-Based and a Statistical-Based Bluff Detection Framework we were able to define and identify bluffing attempts of both agents.
- Both the game theoretic algorithm and the reinforcement learning algorithm exhibit bluffing. However, reinforcement learning (DQN) and game-theoretic (CFR) approaches showed different bluffing strategies: DQN attempts to bluff more conservatively, but since its bluffs are more successful, overall performance is comparable. The response to perceived bluffs shows that both agents act in a very similar way even though they belong to different paradigms.

2 Related Work

Significant progress has been made in the application of AI to games of strategy, and in particular, to poker. Counterfactual Regret Minimization (CFR) has been a foundational algorithm for solving imperfect-information games and has shown remarkable success [17,4]. In 2019 Pluribus achieved superhuman performance in six-player no-limit Texas Hold'em, marking a milestone in multiplayer game AI [1,6]. Simultaneously, Deep Q-Networks (DQN) revolutionized the field of reinforcement learning by combining deep learning with Q-learning to play Atari 2600 games at human level of performance [3]. More recently, Schmid et al. (2023) introduced Student of Games, a unified learning framework capable of operating in both perfect and imperfect-information games [7].

2.1 Leduc Hold'em

Leduc Hold'em is a simplified variant of poker which is commonly used in game theory research because of its small state and action space, making it computationally tractable while retaining essential features of the game [9].

The classic version of Leduc uses a six-card deck (two copies each of King, Queen, and Jack). In this work, however, we extend the game to a full 52-card deck with 13 ranks and four suits, creating richer opportunities for deception. The game is played heads-up over two betting rounds. Each player bets first, with the first player posting one chip and the other player two chips. At the

start, each player receives one private card, hidden from the opponent. Betting follows a fixed-limit structure and players may fold, call, or raise, but the number of raises per round is capped and bet sizes are predetermined.

After the first betting round, a single public card is revealed, followed by a second and final betting round. If neither player folds during the course of the game, a showdown occurs at the end of the second betting round with the winner determined by the following rules: pairs beat high cards and suits serve as tie-breakers (A > K > ... > 2; Spades > Hearts > Diamonds > Clubs). If one player folds, the other immediately wins the pot.

Despite its simplicity, Leduc Hold'em preserves several core strategic features of full-scale poker: hidden information, shifting hand strength after the public card, and the potential for bluffing and deception.

2.2 Bluffing

Bluffing is one of the most iconic aspects of poker. It involves betting high with low cards: 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 [8].

A successful bluff relies not only on bold action, but also on an understanding of how one's 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 [8]. 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. Without bluffing, a player that only bets strong hands becomes exploitable. 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 [2].

Bluffing has been analyzed theoretically using Bayesian probabilistic model that separates the uncertainty of the game dynamic from the uncertainty in an opponents strategy [9]. Their framework enables inference over opponent strategies based on observed actions, allowing the agent to compute a posterior distribution over possible opponent policies. Testing their 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.

2.3 Counterfactual Regret Minimization (CFR)

Counterfactual Regret Minimization (CFR) is a prominent iterative algorithm for computing approximate Nash equilibria in extensive-form games with im-

perfect information [17]. CFR works by simulating repeated plays of the game. CFR tracks for each decision point in the game how much a player "regrets" not having chosen each available action in the past. These regrets are computed using counterfactual values—expected outcomes assuming the player had taken a different action, while the rest of the game proceeds as before [17].

CFR maintains and outputs an average strategy over all iterations, which significantly improves convergence properties [17].

2.4 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 [10,6,3].

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 [13,11]. While tabular Q-learning stores values for each state-action pair, this quickly becomes infeasible in complex environments. DQN addresses this by using a neural network to estimate the Q-value function $Q(s, a; \theta)$, where θ represents the parameters (weights and biases) of the neural network [3]. The network takes a state s as input and outputs an estimated Q-value for each possible action, enabling generalization to unseen states.

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 focused on win rates and exploitability, but not on the qualitative nature of strategies such as bluffing or deception [4]. We address this gap by providing a comparative analysis of bluffing strategies in CFR and DQN agents. The goal is to interpret whether and how bluffing manifests itself in these different algorithmic approaches.

3 Methods

All experiments were conducted using our extended version of Leduc Hold'em, building upon the Python RLCard framework [16]. The RLCard implementation of Leduc Hold'em is structured and was modified in the following ways:

- LeducholdemGame: The main game class which allows gameplay. It manages players, dealer, rounds, and transitions between game states. It was modified to handle a 52-card deck initialization and an expanded state space.
- Player: Represents each agent in the game, stores private hand information, chip count and betting status. It remained unchanged.
- Dealer: Manages card dealing, including assigning private hands and revealing the public card. It was modified to manage a full 52-card deck.
- 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. It was modified to handle the 52-card deterministic hand evaluation. No ties were achieved by using suits as tie-breakers.

The environment uses the following fixed parameters described in Table 1:

Table 1: Leduc Hold'em Environment Parameters

Property	**Description**
Number of players	2 (fixed)
Deck	52 cards (4 suits of each of the 13 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

3.1 Implementation of DQN and CFR

RLCard provides a modular implementation of DQN which is tailored for training agents in imperfect-information card games like poker [16]. The PyTorch-based version closely follows the original DQN algorithm [3] with several practical adaptations to support batch training, evaluation, and integration into the RLCard framework.

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.

To support asymmetric training scenarios and interaction with arbitrary opponent policies, we implemented a custom version of Counterfactual Regret Minimization (CFR), based on RLCards version, that allows CFR to train against a separate, externally defined opponent. 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.

For evaluation and gameplay, the agent selects actions from its average policy, rather than its current iteration policy, since the average policy converges to a more stable and robust strategy over time. 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.

3.2 Simultaneous Training and Evaluation of CFR and DQN

Training DQN and CFR against each other enables both agents to co-evolve their strategies in real time, continuously adapting to the opponent's changing policy.

During the training phase the agents played 100 000 games of Leduc Hold'em against each other. Each training episode consisted of both agents interacting in a shared environment, with CFR always assigned as Player 1 and DQN as Player 0. Before each training episode, CFR performs 10 recursive tree traversal iterations. After a single episode, DQN updates its Q-network using trajectories (data) collected during the episode which allows DQNs strategy to evolve. This creates a continuous feedback loop where both agents adapt to each others evolving strategies. This process continues for 100 000 episode where in each episode 1 game of Leduc Hold'em is played.

The DQN agent uses the following hyperparameters:

Table 2: DQN Hyperparameters

Hyperparameter	**Value**	**Description**
Network Architecture	[256, 256]	Hidden layer sizes (fully connected)
Learning Rate	0.00005	Adam optimizer learning rate
Batch Size	64	Mini-batch size for training
Epsilon Start	1.0	Initial exploration probability
Epsilon End	0.05	Final exploration probability
Epsilon Decay Steps	10,000	Steps for ϵ-greedy exploration decay
Replay Memory Size	20,000	Maximum replay buffer capacity
Replay Memory Init Size	500	Initial experiences before training
Update Target Estimator Every	1,000	Steps between target network updates
Discount Factor (γ)	0.99	Future reward discount factor
Train Every	1	Training frequency (every N steps)
Weight Initialization	Xavier Uniform	Neural network weight initialization
Loss Function	MSE	Mean squared error loss
Optimizer	Adam	Gradient descent optimizer
Activation Function	Tanh	Hidden layer activation function
Batch Normalization	BatchNorm1d	Input layer normalization

After 100 000 training episodes, the trained models were saved, and an evaluation phase was conducted where another 100 000 evaluation games were played with the agents using their learned policies.

3.3 Bluff Detection

First we define the threshold-based bluff detector. To assess whether bluffing occurred, we classify large raise actions as bluff attempts based on private hand

strength and public card information. The hand score formula used is

$$HandScore = \begin{cases} (R_{pc} \times 4) + S_{pc}, & \text{if no pair,} \\ (R_{pc} \times 4) + S_{pc} + 1000, & \text{if there is a pair.} \end{cases} \quad (1)$$

where R_{pc} is the rank of the private card and S_{pc} is the suit of the private card. If an agent performs a raise action while holding a hand with a strength score of 32 or lower (less than 10s and no pairs), then the action is classified as an attempted bluff. If the opponent folds immediately to the agents bluff attempt, then we classify that as a successful bluff. This rule-based detection is used to count bluff attempts during evaluation games.

In addition to the threshold-based bluff detection we also define a statistics-based bluff detection relying on belief distributions and expected values. This definition can be extended to all types of poker.

Let D be the deck, $H \subseteq D$ the set of private hands for a player, and let pc denote the public context (e.g., public card, betting round, position, and other publicly observed features). Let A be the action set (check, call, bet, raise, fold). We write $s(h) \in \mathbb{R}$ as a hand–strength function. It assigns a real number to any hand h and measures how strong that hand is. Two ways of defining $s(h)$ are:

- Showdown equity: The probability of the player winning at showdown against all possible opponent hands.
- Normalized strength index: Assign fixed numbers to hand types, ranked from weakest to strongest. This is how we define $s(h)$ in this paper.

Let $u(h, a) \in \mathbb{R}$ denote the expected utility (EV) of taking action a with hand h at pc. We write $a_{\text{passive}} \in \{check, call\}$ for a non-aggressive alternative available at the same information state. From the perspective of Player i observing Player j take action a at context pc, let

$$\mu(h' \mid a, pc)$$

be Player i's belief distribution over Player j's possible private hands $h' \in H$ after observing a and pc. We use the shorthand $h' \sim \mu(\cdot)$ to denote that h' is a sample from the specified belief distribution. Thus, a player holding h is attempting to bluff with action a at context pc if the action strategically misrepresents strength and is EV-preferred:

$$s(h) \; < \; \mathbb{E}_{h' \sim \mu(h' \mid a, pc)}\big[s(h')\big] \quad \text{and} \quad u(h, a) \; > \; u\big(h, a_{\text{passive}}\big) \quad (2)$$

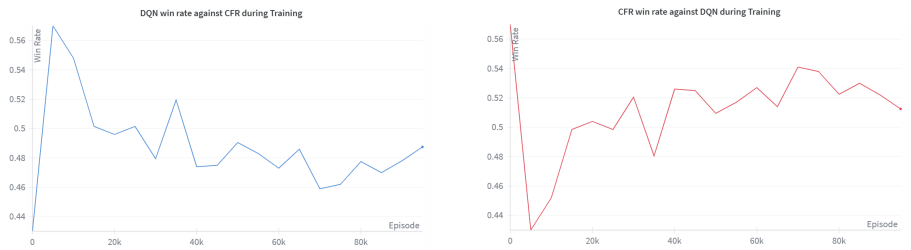
The first clause formalizes misrepresentation: the action a is more typically associated (in the observers beliefs) with stronger hands than the bluffer actually holds. The second clause requires that the bluff is rational (higher EV than the passive alternative). Hand strength $s(h)$ is calculated as it is in the threshold-based bluff detection using the same equation.

Note that our implementation approximates this formal definition, due to the computational intensity of producing the full belief distribution. We compute the

mean and standard deviation of the strengths of hands $s(h)$ that took action a at a decision context. Pair and non-pair cases are kept separate. For pairs, as the misrepresentation clause we use mean $-\ \sigma \times$ threshold where σ is the standard deviation and threshold is 0.5 but it can be modified to fit specific criteria, to give a number which if the actual hand is below this number, then it is classified as misrepresentation. For non-pairs if >70% of raises are with pairs, any non-pair raise satisfies as misrepresentation. Otherwise, we use the same method as in the pair case, but we use numbers from non-pair cases. We use payoffs as EV. Further details of our methods are in [15] and our implementation codebase can be found at [14].

4 Results

In the training phase, both the CFR and DQN agents were trained simultaneously against each other on 100 000 games of Leduc Hold'em with the aim of allowing both agents to adapt to each other.



(a) DQN win rate vs. CFR (b) CFR win rate vs. DQN

Fig. 1: 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 around 50%.

Figure 1 illustrates the win-rate dynamics between the DQN and CFR agents during training. From Figure 1a we can see that initially the DQN agent achieves a slight advantage, with its win rate peaking above 55%. However, this edge quickly diminishes as the CFR agent adapts, which leads to a decline in DQNs win rate which stabilizes in the 46%-49% range for most of the training process. Figure 1b shows the complementary trend, that is, CFR immediately drops from a 56% win rate down to 44%, but it steadily improves after this and maintain a persistent advantage with win rates ranging between 50%-54% during most of the training period.

This observed difference is consistent with the theoretical properties of the two algorithms. Namely, CFR is explicitly designed for imperfect-information games and leverages regret minimization to converge towards equilibrium strate-

gies under the right conditions. Practical convergence to a Nash equilibrium typically requires millions of iterations, and the training budget in this paper was insufficient to guarantee equilibrium play. This can be also seen from the volatile win rate graphs which show that true convergence has not yet been reached.

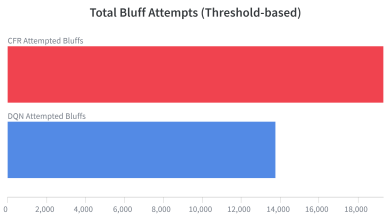
The graphs also indicate partial progress towards less-exploitable play as CFRs average strategy becomes harder to exploit than DQNs function approximation policy. Nevertheless, its regret-minimization updates systematically push strategies closer to equilibrium and eliminate highly exploitable behaviors. By contrast, DQN is optimizing a value function from sampled play against a non-stationary opponent as CFRs policy changes every episode. Off-policy TD methods such as DQN assume a stationary target, however, here that assumption is violated, which amplifies function-approximation noise and over/under estimation. With ϵ-greedy exploration and finite play, DQN probably undersamples rare but crucial situations, so its early strategy gets counter-adapted as CFRs regret updates rebalance action probabilities towards more profitable actions faster.

4.1 Do CFR and DQN bluff?

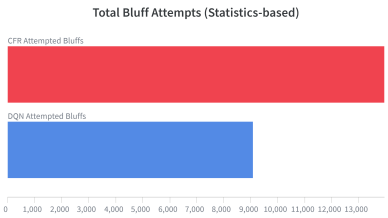
From Figure 2 we can see that both agents have a large number of both attempted and successful bluffs using two independent detectors. This shows us that indeed both DQN and CFR engage in bluffing. The statistics-based detection yields fewer counts than the simpler threshold heuristic, which is to be expected considering that the statistics-based one has a stricter definition of bluffing.

A possible explanation of the differences in bluffing behaviour between the two algorithms is that these differences can be attributed to the paradigms that each agent originates from. CFR attempts to bluff more than DQN because CFR is equilibrium-driven and it must sometimes bluff to remain unpredictable. On the other hand, DQN only bluffs when the estimated Q-value says it is profitable. Since it is not taught about bluffing explicitly, it tends to be more conservative with its attempts. Thus, it under-bluffs compared to CFR.

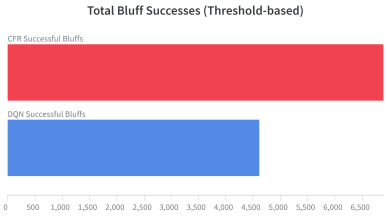
Interestingly, the overall success rates are similar despite the difference in the number of attempts. The bluff success rate for CFR is 36% by the threshold-based detector and 37% by the statistical-based detector, and for DQN it is 34% by the threshold-based detector and 39% by the statistical-based detector. The success rate depends on the willingness of the opponent to fold. Both CFR and DQN learn their folding frequencies during training. CFR learns them as a part of minimizing regret and if folding too often would allow bluffs to exploit it, CFR would reduce folding in those states. If calling too often would lose against value bets, then CFR would increase folding. On the other hand, DQN learns to fold implicitly through its Q-values. If calling has a negative expected value and folding has a higher expected value, the Q-network will assign a higher value to folding. The similar bluff success rate shows that both DQN and CFR on the opponent side have developed a comparable level of strategic competence. This is similar to human poker dynamics where players of similar skill levels will tend



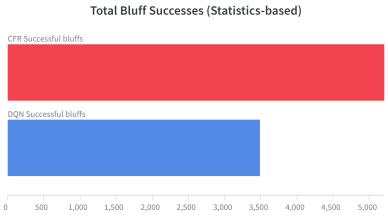
(a) Total number of bluffs attempted by both DQN and CFR (using the threshold-based detector).



(b) Total number of bluffs attempted by both DQN and CFR (using the statistics-based detector).



(c) Total number of Successful bluffs by both DQN and CFR (using the threshold-based detector).

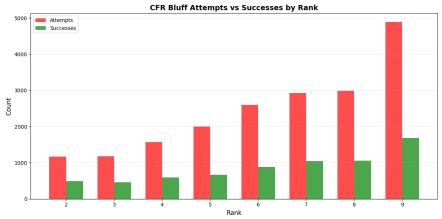


(d) Total number of Successful bluffs by both DQN and CFR (using the statistics-based detector).

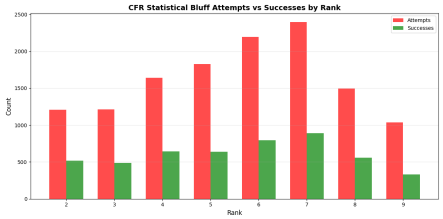
Fig. 2: Bluff attempt and success frequency. Panels (a)–(b) report total bluff attempts under our two bluff detection methods. We can see that in both panels, (a) and (b), CFR attempts more bluffs than DQN. Panels (c)–(d) report the number of bluffs classified as successful under each detector. We can see that CFR has more succesful bluffs than DQN in absolute numbers, however, when taking into account panels (a) and (b) the success rates are very similar.

to achieve similar bluff success rates because their ability to detect and respond to deceptive play is comparably refined.

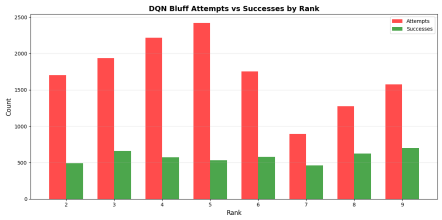
Figure 3a shows that CFRs bluffing attempts increase in size from 2 (the lowest rank) to 9 (the highest rank considered to be a bluff, this is where the cutoff is). The successes roughly scale in the same way, although, with smaller size. This shows that CFR does not only bluff when having a very weak hand, but attempts to bluff also with mid to high-rank cards. CFR's strategy is systematic and integrated into its overall strategy. Most contiguous ranks have similar attempt sizes, which indicates that CFR is preventing to be exploited by distributing its bluffing attempts across many cards. The conversion rate is also roughly the same across most ranks, which indicates that opponents cannot easily exploit CFR's bluffs. The results obtained from the statistics-based detector shown in figure 3b follow the same trend, but with smaller absolute sizes of attempts and successes due to the stricter classification. Both graphs show that CFR bluffs as humans do in real life poker, that is, they choose to bluff with



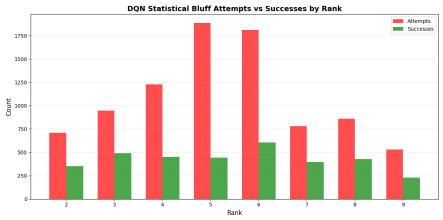
(a) CFR Number of Bluff Attempts and Bluff Successes by Card Rank (using the threshold-based detector).



(b) CFR Number of Bluff Attempts and Bluff Successes by Card Rank (using the statistics-based detector).



(c) DQN Number of Bluff Attempts and Bluff Successes by Card Rank (using the threshold-based detector).

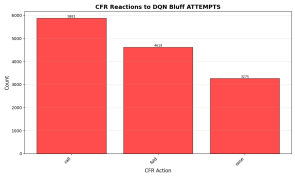


(d) DQN Number of Bluff Attempts and Bluff Successes by Card Rank (using the statistics-based detector).

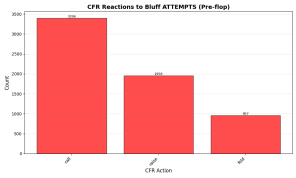
Fig. 3: Bluff attempt and success frequency by card rank. Panels (a)-(b) show the comparison between the number of attempted vs. the number of successful bluffs by CFR under our two detectors. Panel (a) shows that bluff attempts increase with rank, peaking at ranks 7–9 and successes scale proportionally. Panel (b) shows a similar trend, but the attempts peak at ranks 5-7 and then fall off, with successes following a similar pattern. Panels (c)-(d) show the comparison between the number of attempted vs. the number of successful bluffs by DQN under our two detectors. From (c), we can see how attempts scale with rank peaking at ranks 4-5 and then fall off before going up in numbers again. Successes hover around the same number across ranks. From (d), we can see that attempts again scale with rank, peaking at ranks 5-6 and then fall off. The successes hover around the same number but this time with more variation between ranks.

hands that are weak enough to probably lose at showdown but still have some chances of winning as a high card.

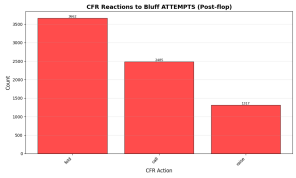
On the other hand, Figure 3c reveals that the DQN agent bluffs most often in the mid-rank region (3-6). These are the hands that are quite weak and very likely to lose at showdown, but are used by the DQN agent to stay in the game through bluffing with them. The success rate is about the same across ranks, but also depends on whether the opponent has learned to associate certain patterns with certain hands and therefore to certain actions such as folding. DQN does not have any built-in concepts such as deception or bluffing. It bluffs only because Q-learning rewards certain aggressive actions that occasionally yield higher ex-



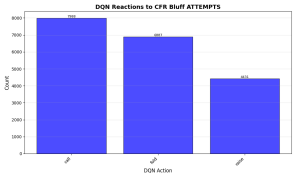
(a) CFR's overall reactions to DQN bluffs, aggregated across all phases (using the threshold-based detector).



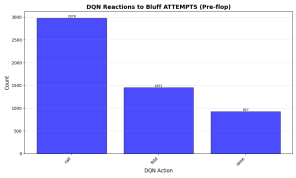
(b) CFR's reactions to DQN bluffs during the pre-flop stage (using the threshold-based detector).



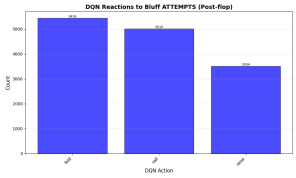
(c) CFR's reactions to DQN bluffs during the post-flop stage (using the threshold-based detector).



(d) DQN's overall reactions to CFR bluffs, aggregated across all phases (using the threshold-based detector).



(e) DQN's reactions to CFR bluffs during the pre-flop stage (using the threshold-based detector).

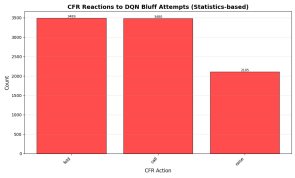


(f) DQN's reactions to CFR bluffs during the post-flop stage (using the threshold-based detector).

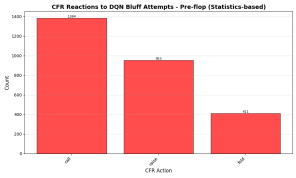
Fig. 4: Opponent reactions to bluffing under the threshold-based detector. Panel (a) highlights how frequently CFR chooses to fold, call, or raise in response to DQN's bluff attempts. Panels (b)-(c) show the same just across the two different phases of the game. CFR prefers to call overall and in the pre-flop stage, while in the post-flop stage it prefers to fold. Panel (d) reveals how often DQN chooses to fold, call, or raise in response to CFR's bluff attempts. Panels (e)-(f) show the same just across the two different stages of the game. DQN prefers to call overall and in the pre-flop stage, while preferring to fold in the post-flop stage. Both agents react similarly.

pected return than folding outright. The possible reason why the bluffing clusters in the mid-rank zone is because the risk-reward trade-off has showed up the most in these zones in its experience. Additionally, training against CFR also shapes its bluffing style. CFR is trying to minimize its own exploitability which prevents DQN from becoming an all-in-bluffer because DQN would lose badly. Instead, it pushes DQN to find a more selective group of hands where bluffing still works against CFR. Figure 3d shows a very similar trend, the biggest difference being the smaller quantity of games due to stricter classification.

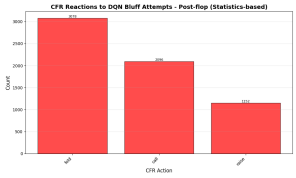
Figure 4a shows that the most common action of CFR to DQNs bluffing attempts is calling, followed by folding and raising. This indicates that CFR chooses to see through the bluff rather than retreat (fold) or counterattack (reraise). This might be done in order to minimize regret by avoiding to fold too early. CFR folds when the expected risk is too high to avoid leaking chips. It might raise when it is extremely confident that the opponent is bluffing in order



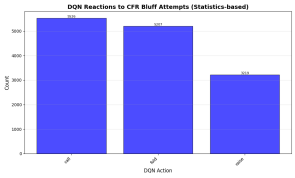
(a) CFR's overall reactions to DQN bluffs, aggregated across all phases (using the statistics-based detector).



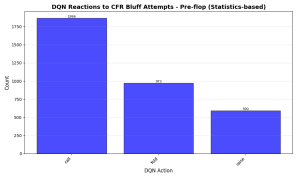
(b) CFR's reactions to DQN bluffs during the pre-flop stage (using the statistics-based detector).



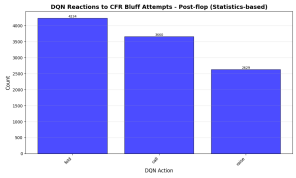
(c) CFR's reactions to DQN bluffs during the post-flop stage (using the statistics-based detector).



(d) DQN's overall reactions to CFR bluffs, aggregated across all phases (using the statistics-based detector).



(e) DQN's reactions to CFR bluffs during the pre-flop stage (using the statistics-based detector).



(f) DQN's reactions to CFR bluffs during the post-flop stage (using the statistics-based detector).

Fig. 5: Opponent reactions to bluffing under the statistics-based detector. Panel (a) highlights how frequently CFR chooses to fold, call, or raise in response to DQN's bluff attempts. Panels (b)-(c) show the same just across the two different phases of the game. CFR prefers to call and fold overall. Furthermore, in the pre-flop stage it prefers to call, while in the post-flop stage it prefers to fold. Panel (d) reveals how often DQN chooses to fold, call, or raise in response to CFR's bluff attempts. Panels (e)-(f) show the same just across the two different stages of the game. DQN prefers to call overall and in the pre-flop stage, while preferring to fold in the post-flop stage. Both agents react very similarly.

to make the opponent fold. To get a more nuanced understanding of the reactions, we also analyze the reactions per game phase. Figure 4b reveals that before the public card is shown CFR prefers to be in the game and gather more information rather than to fold prematurely. It also likes to reraise, possibly to scare off the opponent. Lastly, it folds the least. However, when looking at the post-flop phase from figure 4c, once the public card is revealed, CFR prefers to fold rather than to raise or call. This indicates that CFR becomes more conservative once more information is available, choosing to fold in unfavorable situations or when DQNs bets appear to be strong. The statistics-based detector produces similar results as can be seen in Figures 5a, 5b and 5c. The statistics-based detector is used to verify that the general patterns are still there even when a different way of detecting bluffs is used. The main difference is that there is less cases in the statistics-based detector graphs. Folding also becomes the main overall action, but since it is only by a few cases, it can be plausibly attributed to chance.

DQN follows a similar pattern as CFR, preferring to call most of the bluffing attempts, then to fold and then to raise. This can be seen from Figure 4d which follows the same pattern as CFR's reactions but with more cases. It is not surprising that DQN prefers to call the most in its response since calling is often the least punishing action because it allows the agent to gather more information about outcomes (to go and see showdowns). Folding would cut off rewards entirely (as you give up and do not see your opponents cards) and raising would introduce higher variance that could destabilize value estimation. DQN folds only when the learned Q-values suggest that continuing would result in negative expected reward, just as humans throw away hopeless hands in real life poker. Similarly, humans raise only when they are confident that they hold either a really strong hand or they are confident that the opponent is bluffing or they know that by raising they could make the opponent fold, and DQN behaves in a similar way where its more risk-averse and avoids over-raising. DQNs call-heavy response pattern shows that it has learned a conservative, information-gathering strategy rather than an exploitative one. When we split the responses per game phase, we can see, from Figure 4e, that in the pre-flop stage, DQN still behaves in a information-gathering strategy because the public card has not been revealed and uncertainty is high. Calling is a natural middle-ground because it keeps options open without risking too much. Folding would waste potential equity, as even weak hands can sometimes improve once the public card is revealed due to forming a pair for example. However, once the public card is revealed there is not a lot of additional information to be gained, and DQNs primary response becomes folding. This is similar to how humans play poker in that they fold more in post flop because bluffs are harder to spot with community cards being revealed. DQN shifts to a much more conservative strategy post-flop, preferring to cut losses (fold) over information-gathering (call).

5 Conclusions and Further Research

We explore bluffing behavior in Leduc Hold'em by training and evaluating two fundamentally different agents, namely, reaction based Deep Q-Networks (DQN) versus forward-looking game-theoretic Counterfactual Regret Minimization (CFR). We find that both agents are capable of bluffing.

We have trained and evaluated DQN and CFR. CFR slightly dominated DQN in win rate which is to be expected since CFR computes the Nash-theoretic equilibrium, which is guaranteed to be unexploitable, on average. Interestingly, both agents exhibited bluffing behavior even though they are based on different principles. Neither agent had been taught deception and bluffing explicitly but bluffing emerged because of the training paradigms, rules of the game and because of each other. CFR attempted more bluffs than DQN but both ended up having roughly the same bluffing success rate. Looking at the bluffing styles of both agents, CFR preferred to use mid-strength hands to bluff, while DQN used more low-strength hands.

For the reactions of both agents to each others bluffing attempts, both followed the same patterns of being exploratory and risk-minimizing. In the pre-flop phase they remained exploratory while in the post-flop phase they became more conservative once most of the possible information about that round was gathered. Both agents developed a similar bluffing and response pattern, which is surprising since they are based on different paradigms. Despite this difference, they developed similar behavior that is closely linked to how humans play poker.

5.1 Limitations and Further Research

This work has the following limitations. First, all experiments were conducted in the 52-card version of Leduc Hold'em, a simplified version of real Texas Hold'em regarding the number of players, number of public cards, number of private cards, and betting amounts. It would be interesting to see when more than two players are present, if the agents target specifically some agents with the bluffs, perhaps the weakest links, or do they attempt to bluff everyone equally. Additionally, in No Limit poker it would be interesting to see how the agents decide the betting amount when bluffing.

Second, due to practical resource limitations, especially in regard to training time and computational power, only a finite number of training episodes and evaluation games could be executed. It is possible that more training episodes or alternative configurations of the hyperparameters could yield stronger or more refined agents and behaviors. Potential future work could try to run a hyperparameter sweep [12] to make the agents more stable or perhaps make the agents bluff more/less.

Third, 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. Future work could look into using different versions of these algorithms and seeing how they perform.

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

References

1. Brown, N., Sandholm, T.: Superhuman ai for multiplayer poker. Science **365**(6456), 885–890 (2019). https://doi.org/10.1126/science.aay2400, https://doi.org/10.1126/science.aay2400
2. Chen, B., Ankenman, J.: The Mathematics of Poker. ConJelCo LLC, Pittsburgh, PA (2006)
3. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis,

D.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015). https://doi.org/10.1038/nature14236, https://doi.org/10.1038/nature14236
4. Moravčík, M., Schmid, M., Burch, N., Lisý, V., Morrill, D., Bard, N., Davis, T., Waugh, K., Johanson, M., Bowling, M.: Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. Science **356**(6337), 508–513 (2017). https://doi.org/10.1126/science.aam6960
5. von Neumann, J., Morgenstern, O.: Theory of Games and Economic Behavior. Princeton University Press, Princeton, NJ (1944)
6. Plaat, A.: Learning to play: reinforcement learning and games. Springer (2020)
7. Schmid, M., Moravcik, M., Burch, N., Kadlec, R., Davidson, J., Waugh, K., Bard, N., Timbers, F., Lanctot, M., Holland, G.Z., Davoodi, E., Christianson, A., Bowling, M.: Player of games. CoRR **abs/2112.03178** (2021), https://arxiv.org/abs/2112.03178
8. Sklansky, D.: The Theory of Poker. Two Plus Two Publishing, Las Vegas, NV (1987)
9. Southey, F., Bowling, M.P., Larson, B., Piccione, C., Burch, N., Billings, D., Rayner, C.: Bayes' bluff: Opponent modelling in poker. In: Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI). pp. 550–558 (2005)
10. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, 2nd edn. (2018)
11. Wang, H., Emmerich, M., Plaat, A.: Assessing the potential of classical q-learning in general game playing. In: Benelux Conference on Artificial Intelligence. pp. 138–150. Springer (2018)
12. Wang, H., Emmerich, M., Preuss, M., Plaat, A.: Hyper-parameter sweep on alphazero general. arXiv preprint arXiv:1903.08129 (2019)
13. Watkins, C.J., Dayan, P.: Q-learning. In: Machine Learning. vol. 8, pp. 279–292. Springer (1992)
14. Začiragić, T.: Bluffing by DQN and CFR in Leduc Hold-em Poker: Codebase. https://github.com/TarikZ03/Bluffing-by-DQN-and-CFR-in-Leduc-Hold-em-Poker-Codebase (2025), gitHub repository
15. Začiragić, T.: Bluffing tendencies in Leduc Hold'em. Leiden University (2025)
16. Zha, D., Lai, K., Cao, Y., Huang, S., Wei, R., Guo, J., Hu, X.: Rlcard: A toolkit for reinforcement learning in card games. CoRR **abs/1910.04376** (2019), http://arxiv.org/abs/1910.04376
17. Zinkevich, M., Johanson, M., Bowling, M., Piccione, C.: Regret minimization in games with incomplete information. In: Proceedings of the 21st International Conference on Neural Information Processing Systems. p. 1729–1736. NIPS'07, Curran Associates Inc., Red Hook, NY, USA (2007)