Self-Play Reinforcement Learning Without Tree Search in Connect Four

Umer Raja
Department of Computer Science
LUMS
Lahore, Pakistan
26100063@lums.edu.pk

Haris Rathore
Department of Computer Science
LUMS
Lahore, Pakistan
26100064@lums.edu.pk

Abstract

The remarkable success of AlphaZero demonstrated that combining deep neural networks with Monte Carlo Tree Search (MCTS) yields superhuman performance in complex board games. However, the dependence on large-scale search introduces substantial computational overhead, limiting applicability in resource-restricted environments. This paper explores an alternative approach: training model-free agents via self-play without explicit lookahead search. We focus on Connect Four, modeling it as an Egocentric Markov Decision Process (E-MDP). We conduct a comparative analysis of three distinct reinforcement learning algorithms: Proximal Policy Optimization (PPO), Deep Q-Networks (DQN), and REINFORCE. To overcome the instability inherent in non-stationary self-play, specifically the emergence of reactive strategy bias and cycling, we implement Fictitious Self-Play (FSP) and evaluate the impact of heuristic reward shaping. Our results from a round-robin tournament demonstrate that Policy Gradient methods (PPO and REINFORCE) significantly outperform DQN in this domain. While reward shaping accelerates convergence, agents trained with sparse rewards and FSP achieve comparable strategic ceilings, consistently defeating baseline opponents and showing robust defensive capabilities against historical versions of themselves.

1 Introduction

The mastery of combinatorial board games has long been a benchmark for Artificial Intelligence. The landmark success of AlphaZero demonstrated that a generalized reinforcement learning (RL) algorithm could achieve superhuman performance across Chess, Shogi, and Go without domain-specific heuristics. However, AlphaZero and its successors rely heavily on Monte Carlo Tree Search (MCTS), a lookahead planning algorithm, to guide the neural network's training and inference. While effective, MCTS introduces significant computational overhead, rendering such agents unsuitable for real-time applications or resource-constrained environments where decisions must be made instantly based on "intuition" (policy) rather than exhaustive calculation.

This project explores the capabilities of model-free Deep Reinforcement Learning in the absence of explicit tree search. We focus on Connect Four, a zero-sum, perfect-information game with a state-space complexity of approximately 4.5×10^{12} positions. While Connect Four is theoretically solved, it remains a challenging testbed for learning tactical, spatial strategies purely from self-interaction.

Training agents via self-play without a search-based supervisor presents unique challenges. First, the learning environment is non-stationary; as the agent improves, the opponent (itself) also changes, creating a "moving target" for the optimization landscape. Second, naive self-play often suffers from cycling (e.g., Rock-Paper-Scissors dynamics), where an agent learns to exploit specific weaknesses of its current policy but forgets how to defend against earlier, simpler strategies. In our initial

Preprint. Under review.

experiments, we also observed a reactive strategy bias, where the agent converged to a passive, defensive local optimum, allowing the second player (Player 2) to dominate by simply exploiting the first player's over-extensions.

To address these instabilities, we propose a robust training pipeline applicable across various model-free paradigms, including Proximal Policy Optimization (PPO), Deep Q-Networks (DQN), and REINFORCE. Our unified approach employs an Egocentric Markov Decision Process (E-MDP), enabling a single neural network to generalize across both player positions by transforming the board state relative to the current player's perspective. Furthermore, we mitigate cycling and reactive bias by implementing Fictitious Self-Play (FSP), maintaining a history pool of past policy versions. By training the agent against a mixture of its latest self and historical snapshots, we force the emergence of robust, proactive strategies that generalize across varying skill levels. This work culminates in a comparative tournament to precisely map the performance hierarchy of these model-free methods when operating under a shared, stabilized self-play framework.

2 Literature Review

In recent years, there has been growing interest in applying deep reinforcement learning (RL) techniques directly to the game of Connect Four, particularly in settings that do not rely on explicit tree search. This line of research aims to investigate how far self-play alone can push learning-based agents toward expert-level performance. A number of studies have explored the use of popular deep RL algorithms, such as Deep Q-Networks (DQN), policy gradient methods, and actor–critic approaches, to train Connect Four agents through self-play. In this section, we review several representative works and highlight key findings that motivate the methodological choices in this project.

2.1 Deep RL vs Search: Experimental Comparisons

One experimental comparison is presented by Dabas et al. (2022), who trained three different agents to play Connect Four using distinct paradigms: (1) a minimax algorithm with alpha–beta pruning, (2) a Monte Carlo Tree Search (MCTS) agent, and (3) a deep Q-learning agent based on Double DQN, trained via self-play. In order to ensure fairness, each agent was allocated comparable computational resources in terms of decision time or number of simulations per move. The outcomes of head-to-head evaluations were revealing. The MCTS agent emerged as the strongest overall performer, consistently defeating the deep Q-learning agent, while the deep Q agent itself reliably outperformed the minimax agent under these constraints [9].

Interestingly, the minimax agent was ranked last in this experimental setup, which may initially seem counterintuitive given that Connect Four is a solved game under perfect play. However, this result can be explained by the fact that the minimax agent was deliberately limited in search depth in order to comply with the imposed computational budget. As noted by the authors, the minimax algorithm "was not able to explore higher depth in the game tree" within the allotted time [10]. This restriction allowed learning-based agents to exploit tactical weaknesses arising from shallow search. Over the course of self-play training, the Double DQN agent improved steadily and eventually surpassed the constrained minimax player, demonstrating that a neural network can acquire effective Connect Four tactics capable of beating a non-optimized search strategy. Nevertheless, the MCTS agent remained superior, defeating the DQN agent in the majority of games [9]. Overall, this hierarchy (MCTS > Deep RL > limited-depth minimax) suggests that, under equal computational budgets, learning-based methods can outperform weak search, but well-resourced search techniques such as MCTS still maintain a competitive edge.

2.2 Self-Play RL Performance

A broader comparison is provided by Taylor and Stella [11], who evaluated multiple approaches for playing Connect Four, including a model-free Q-learning agent, a pure MCTS player, and a classical minimax agent, along with several advanced variants of each [14]. Their experimental results showed that, when tested under consistent conditions, the MCTS-based agent achieved the highest overall win rate, followed by the minimax (alpha–beta) agent, with the model-free RL agent performing

worst among the three [15]. While the Q-learning agent augmented with function approximation was able to learn a reasonable level of play, it did not match the performance of the search-based methods.

One notable advantage of the RL approach, however, was computational efficiency at inference time. Because action selection for the trained Q-learning agent requires only a forward pass through a neural network, decisions were made almost instantaneously. In contrast, both MCTS and minimax required significantly more computation per move due to repeated simulations or search expansions. Taylor and Stella report that the Q-learning agent was the fastest to act, whereas MCTS and minimax, despite their superior playing strength, were slower because of this overhead [16]. This highlights an important trade-off between runtime efficiency and playing strength: while pure RL agents are computationally cheap at deployment, they may not achieve the same level of performance as search-based approaches.

Taylor and Stella also reference findings from Scheiermann and Konen [10], who investigated a hybrid AlphaZero-style approach that combines a neural network with an MCTS wrapper. In Connect Four and other board games, this hybrid method was shown to outperform both pure RL and pure search agents. For example, an AlphaZero-inspired agent using a neural network-guided MCTS convincingly defeated both a stand-alone RL agent and a basic MCTS agent in direct competition [9]. In one reported experiment, a vanilla MCTS agent with 10,000 playouts achieved a 100% win rate against a pure RL agent, while the network-assisted MCTS achieved a 99% win rate against the basic MCTS agent [9]. These results further reinforce the conclusion that combining learned representations with search yields the strongest performance, while pure RL agents remain the weakest when evaluated in isolation.

2.3 Improving RL Agents: Reward Shaping

A major challenge when applying pure reinforcement learning to games such as Connect Four is reward sparsity. In standard formulations, the agent receives a meaningful reward only at the end of the game, corresponding to a win or a loss, despite the fact that games may consist of many intermediate moves. To alleviate this issue, several studies have explored reward shaping techniques that provide intermediate feedback to guide learning. Common approaches include assigning small positive rewards for partial achievements, such as forming two-in-a-row or three-in-a-row patterns, or for occupying strategically important positions like the center column, which are heuristically known to correlate with winning chances [11].

Provided that such shaping rewards are potential-based (meaning they can be derived from a potential function over states) they should, in theory, preserve the optimal policy while accelerating learning. Taylor and Stella [11] describe a shaped reward scheme used in prior literature, where agents receive incremental rewards such as +1 for each own piece in a row, +2 for two in a row, and +6 for three in a row, along with a large terminal reward for a win (+40) and a significant penalty for a loss (-30), supplemented by smaller bonuses. By enriching the feedback signal in this way, an RL agent can more rapidly learn important tactical concepts, such as the value of extending a three-in-a-row, compared to learning solely from terminal outcomes. Empirical evidence suggests that reward shaping, combined with techniques such as decaying exploration rates and adaptive learning rates, can substantially improve training efficiency for Q-learning agents in Connect Four [12]. Nevertheless, reward shaping must be applied carefully to avoid introducing misleading incentives; ideally, shaped rewards should reflect genuine progress toward winning. In practice, reward shaping and related techniques, including curriculum learning and self-play against progressively stronger opponents, have proven essential for achieving reasonable training times in such domains.

2.4 Deep Q-Network (DQN)

Deep Q-Networks represent a value-based approach in which a neural network is used to approximate the action-value function for each state–action pair. In the context of Connect Four, a DQN takes the current board position as input and outputs Q-values for each of the seven possible column moves. Training is typically carried out through self-play, with the agent using an ϵ-greedy policy that mostly selects the action with the highest estimated Q-value while occasionally exploring random legal moves. Over time, the agent updates its Q-values based on observed game outcomes.

One challenge faced by DQN in self-play settings is the non-stationary nature of the environment, since the opponent is effectively a continually changing version of the agent itself. Experience replay

is commonly employed to address this issue by training the network on a buffer of past experiences, thereby reducing correlations between updates. In addition, the use of a separate target network can help stabilize learning and prevent oscillatory behavior. In practice, extensions such as Double DQN are often adopted to mitigate the overestimation of Q-values. Previous work has shown that DQN-based agents are capable of learning reasonably strong Connect Four strategies [9]. However, these agents may converge slowly and are susceptible to becoming trapped in suboptimal policies if exploration is insufficient or if rewards are overly sparse. Moreover, standard Q-learning does not explicitly model the opponent's optimal responses, even though Connect Four is a two-player zero-sum game where anticipating best play from the opponent is crucial. This limitation becomes particularly relevant in self-play dynamics, which are discussed further below.

2.5 Policy Gradient (REINFORCE)

Policy-based methods take a different approach by learning a direct mapping from states to a probability distribution over actions, without explicitly estimating state–action values. One of the simplest such methods is REINFORCE, which updates the policy parameters in a direction that increases the likelihood of actions that lead to wins and decreases the likelihood of actions that lead to losses, using the final game outcome as the reward signal. In the context of Connect Four, a policy network, typically a convolutional neural network to accommodate the grid-based input, produces a probability distribution over the seven possible moves. Through self-play, REINFORCE gradually biases the policy toward actions that are more frequently associated with winning trajectories.

An advantage of policy gradient methods is that they directly optimize the objective of winning games and naturally support stochastic policies. However, REINFORCE is well known for the high variance of its gradient estimates, as the outcome of a single game assigns credit or blame to many preceding actions. As a result, it can be difficult for the algorithm to identify which specific decisions were responsible for success or failure. To address this issue, baseline subtraction, often using a value function estimate, is commonly applied to reduce variance. While pure policy gradient methods are capable of learning strong strategies, they typically require a large number of training samples and may be less sample-efficient than value-based or actor-critic methods in a domain such as Connect Four.

2.6 Actor–Critic and PPO

Actor–critic methods combine elements of both policy-based and value-based learning. In a Connect Four setting, an actor–critic agent consists of a policy network (the actor), which proposes moves, and a value network (the critic), which estimates the expected outcome from a given board position. During training, the critic's estimate is used to compute an advantage signal that indicates how much better or worse a taken action was compared to the expected value. The policy is then updated to favor actions with positive advantage. This structure generally leads to more stable and efficient learning than REINFORCE, as the critic provides a denser learning signal at each move.

3 Problem Formulation

We model the game of Connect Four as a finite, two-player, zero-sum Markov Game, formally defined by the tuple $\mathcal{G} = \langle \mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$.

3.1 Game Dynamics

- **Players ($\mathcal{N}$):** The set of players $\mathcal{N} = \{1, 2\}$.
- **State Space ($\mathcal{S}$):** The game is played on a grid with $H = 6$ rows and $W = 7$ columns. A raw state S_{raw} is represented as a tensor $B \in \{-1, 0, 1\}^{H \times W}$, where 1 represents Player 1's pieces, -1 represents Player 2's pieces, and 0 represents empty cells.
- **Action Space ($\mathcal{A}$):** The action space is discrete, corresponding to the column indices $\mathcal{A} = \{0, 1, \ldots, 6\}$. An action $a \in \mathcal{A}$ is valid in state s if and only if the chosen column is not full (height $< H$). We denote the set of legal actions as $\mathcal{A}_{legal}(s)$.

- **Transition ($\mathcal{T}$):** The transition function $\mathcal{T}(s, a) \rightarrow s'$ is deterministic. Upon selecting column a, the piece is placed in the lowest available row r in that column. The turn then passes to the opponent.

3.2 Egocentric MDP Formulation

Standard multi-agent RL often trains separate policies for each player. To enable efficient Self-Play using a single set of parameters θ, we formulate the problem as an Egocentric MDP.

We define an observation transformation function $\phi(S_{raw}, p)$ that orients the board relative to the current player p:

$$\phi(S_{raw}, p) = \begin{cases} S_{raw} & \text{if } p = 1 \\ -S_{raw} & \text{if } p = 2 \end{cases}$$

Under this transformation, the neural network input is always normalized such that the "Agent's Pieces" are represented by positive values (Channel 0) and "Opponent's Pieces" by negative values (Channel 1).

3.3 Reward Structure

We use a hybrid reward design intended to speed up learning by combining sparse terminal outcomes with denser, tactically meaningful shaping signals. At each timestep, the agent receives a reward

$$r_t \;=\; r_{\text{terminal}}(s_t, a_t, s_{t+1}) \;+\; r_{\text{shape}}(s_t, a_t, s_{t+1}).$$

Terminal Rewards. The terminal component r_{terminal} is defined as:

$$r_{\text{terminal}} \;=\; \begin{cases} +40 & \text{if the agent wins (i.e., creates a connect-four)} \\ -30 & \text{if the episode ends in a loss or an illegal move} \\ 0 & \text{if the game ends in a draw} \\ 0 & \text{if the state is non-terminal.} \end{cases}$$

Because the environment alternates turns, a loss is typically realized on the opponent's move. To preserve correct credit assignment under self-play, we apply the -30 loss penalty to the losing player's final action at the learner level (rather than relying on the environment to emit the loss on the opponent's turn).

Shaping Rewards. To provide intermediate learning signals, we add a shaping term that encourages tactically strong play. In particular, we reward the creation of aligned streaks (and optionally central control), following a commonly used Connect Four shaping scheme:

$$r_{\text{shape}} \;=\; r_{\text{line}} \;+\; r_{\text{center}}.$$

- **Line/Streak Shaping (r_{line}):** After the agent plays, we evaluate newly formed aligned configurations and assign:
  - $+1$ for each one-in-a-row contribution,
  - $+2$ for forming a two-in-a-row,
  - $+6$ for forming a three-in-a-row,

  where these values are computed over all relevant directions (horizontal, vertical, and diagonal) for the current player.
- **Centrality Shaping (r_{center}):** A small bonus is assigned for playing closer to the center columns, reflecting the empirical importance of central control in Connect Four. The bonus is implemented as a fixed, column-dependent reward added on the step the piece is placed.

Total Reward. The reward used for learning is therefore

$$r_t \;=\; r_{\text{terminal}} \;+\; r_{\text{line}} \;+\; r_{\text{center}},$$

which maintains a strong terminal objective (winning) while providing denser guidance for tactical and positional play. Although the shaping term introduces non-zero intermediate rewards, the overall training remains adversarial through self-play, and terminal outcomes continue to dominate the optimization signal.

4 Methodology

We study three reinforcement learning paradigms for learning Connect Four strategies under self-play: (i) Proximal Policy Optimization (PPO), (ii) Deep Q-Networks (DQN), and (iii) REINFORCE. All agents operate within a zero-sum, deterministic, turn-based environment and are trained under a shared self-play framework.

4.1 State Representation and Environment

The environment is modeled as a Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ with alternating turns. A rotated observation scheme is used so that the learning agent always perceives itself as the current player.

- **State s_t:** A tensor of shape $(3, 6, 7)$:
  - Channel 0: current player's pieces
  - Channel 1: opponent's pieces
  - Channel 2: broadcasted legal move mask
- **Action space $\mathcal{A}$:** Discrete set of 7 column indices.
- **Reward function:** Terminal rewards are $+40$ for a win, -30 for a loss or illegal move, and 0 for a draw. Intermediate reward shaping is applied based on streak formation and centrality heuristics.

All agents share the same environment dynamics and reward structure to ensure fair comparison.

4.2 Neural Network Architecture

For PPO and REINFORCE, we employ a shared convolutional neural network architecture:

- **Backbone:** Three convolutional layers with $(32, 64, 64)$ filters, 3×3 kernels, stride 1, and ReLU activations.
- **Shared Dense Layer:** A fully connected layer with 512 hidden units.
- **Heads:**
  - **Actor $\pi_\theta(a|s)$:** Softmax distribution over 7 actions.
  - **Critic $V_\phi(s)$:** Linear scalar output estimating expected return.

For DQN, the same convolutional backbone is used, followed by a linear head outputting a 7-dimensional vector of state-action values $Q_\theta(s, a)$.

4.3 Proximal Policy Optimization (PPO)

PPO is used as the primary on-policy actor–critic baseline. Policy updates are constrained via clipped probability ratios to stabilize training.

4.3.1 Objective Function

The PPO objective is defined as:

$$\mathcal{L}^{\text{PPO}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right]$$

where

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}.$$

4.3.2 Action Masking

Illegal actions are masked by setting their logits to $-\infty$ prior to softmax:

$$\pi_\theta(a_{illegal}|s) = 0.$$

4.3.3 Zero-Sum Generalized Advantage Estimation

To account for alternating turns and perspective flips, GAE is modified as:

$$\delta_t = r_t + \gamma(-V(s_{t+1})) - V(s_t),$$

ensuring that opponent value estimates are negated in accordance with zero-sum dynamics.

4.4 Fictitious Self-Play (FSP)

To mitigate non-stationarity and cycling behaviors, we adopt Fictitious Self-Play.

A population buffer $\mathcal{P}$ of size $N = 20$ stores historical snapshots of the policy. During training, opponents are sampled as:

- 80%: current policy π_{current}
- 20%: uniformly sampled historical policy $\pi_{\text{old}} \sim \mathcal{P}$

During optimization, gradients are masked such that the agent only updates parameters based on its own actions; opponent actions are treated as part of the environment dynamics.

4.5 Deep Q-Networks (DQN)

DQN is used as an off-policy value-based baseline.

4.5.1 Bellman Update

The action-value function $Q_\theta(s, a)$ is trained by minimizing the temporal-difference loss:

$$\mathcal{L}^{\text{DQN}}(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(r + \gamma \max_{a'} Q_{\theta^-}(s', a') - Q_\theta(s, a)\right)^2\right],$$

where θ^- denotes parameters of a target network updated periodically.

4.5.2 Experience Replay and Masked Greedy Policy

Transitions are stored in a replay buffer $\mathcal{D}$ to decorrelate updates. Action selection follows an ϵ-greedy strategy with illegal actions masked:

$$a_t = \begin{cases} \text{random legal action}, & \text{with probability } \epsilon, \\ \arg\max_{a\in\mathcal{A}_{legal}} Q_\theta(s_t, a), & \text{otherwise.} \end{cases}$$

4.5.3 Loss Assignment for Losing Player

Since losses occur on the opponent's turn, a terminal loss penalty of -30 is retroactively applied to the losing player's final action to preserve correct credit assignment in self-play.

4.6 REINFORCE

REINFORCE is used as a Monte Carlo policy-gradient baseline without a value function.

4.6.1 Policy Gradient Objective

The policy parameters θ are updated via:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\, G_t\right],$$

where $G_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$ is the discounted return.

4.6.2 Terminal Credit Assignment

For episodes ending in a normal win, a loss penalty of -30 is applied to the losing player's final action prior to return computation, aligning REINFORCE with the zero-sum reward structure.

4.7 Pseudocode Summary

Algorithm 1 Self-Play Training Loop

```
for each episode do
    Sample opponent from FSP pool
    Roll out trajectory under rotated observations
    if algorithm = PPO then
        Compute GAE with zero-sum correction
        Update actor–critic via clipped objective
    else if algorithm = DQN then
        Store transitions in replay buffer
        Update Q_θ via TD loss
    else if algorithm = REINFORCE then
        Compute Monte Carlo returns
        Apply policy-gradient update
    end if
    Periodically evaluate vs random and self-play
end for
```

5 Experiments

5.1 Experimental Setup

We evaluated three distinct reinforcement learning paradigms under a unified self-play environment: (i) Proximal Policy Optimization (PPO), (ii) Deep Q-Networks (DQN), and (iii) REINFORCE. All agents utilize the same three-layer CNN backbone architecture. Training was performed using 16 parallel environments to generate a high volume of interaction data.

PPO Training Parameters The PPO agents were trained for 2 million timesteps. Key hyperparameters included a learning rate of 2.5×10^{-4}, an entropy coefficient of 0.03, a γ of 0.99, a GAE λ of 0.95, and a total batch size of 8192 (512 steps $\times$ 16 envs).

DQN Training Parameters The DQN agent was trained for 1.5 million timesteps, chosen to roughly match the PPO agent's sample efficiency in terms of total environment steps. The learning rate was set to 1.0×10^{-4}, the replay buffer size to $200,000$, and the batch size to 256. A crucial setting was the ϵ-greedy exploration schedule, which decayed linearly to a final ϵ of 0.05 over 30% of the total training steps. The target network was updated every $2,000$ timesteps.

REINFORCE Training Parameters The REINFORCE agent (Monte Carlo policy gradient) was trained for $30,000$ episodes. Due to its high-variance nature, this agent required episodic returns, making timesteps an inappropriate comparison metric. Hyperparameters included a learning rate of 1.0×10^{-4}, a γ of 0.99, and an entropy coefficient of 0.01.

5.2 Convergence Behavior

Initial experiments with Vanilla PPO (Latest vs Latest) and a low entropy coefficient (0.01) resulted in a strong bias towards Player 2, achieving win rates $\approx 80\%$. We hypothesize this occurred because reactive blocking strategies are easier to learn than proactive trap construction. This necessitated the implementation of Fictitious Self-Play (FSP) for all algorithms.

Policy Gradient (PPO/REINFORCE) Upon increasing the PPO entropy coefficient to 0.03 and implementing FSP, we observed stabilized learning. The Critic loss converged to ≈ 0.1, indicating accurate value estimation, while the Actor loss exhibited a U-shaped curve, reflecting the transition from learning basic rules to optimizing against a skilled adversary. REINFORCE displayed higher variance in its gradient but showed steady improvement in its win rate when evaluated against fixed baselines.

Value-Based (DQN) The DQN agent converged much more slowly, with the Q-values showing significant oscillation throughout training. The non-stationary environment proved particularly challenging, requiring the large replay buffer and frequent target network updates to prevent divergence. Despite these stabilizers, the final performance was notably lower than the Policy Gradient methods.

5.3 Head-to-Head Tournament

To rigorously assess relative strength beyond simple baselines, we conducted a round-robin tournament among the trained agents. The pool of agents included the three PPO variants (Vanilla, FSP, Dense), a Deep Q-Network (DQN) baseline, and two variations of REINFORCE (a manual implementation and a library-based implementation via Tianshou).

In each matchup (Agent A vs. Agent B), a series of games was played where both agents alternated between starting as Player 1 and Player 2. This setup ensures that the results account for the inherent First-Player Advantage (FPA) in Connect Four. We report the win rates for both sides to analyze not just dominance, but also defensive capabilities when playing second.

6 Results

We evaluated six trained agents under the shared self-play framework: (1) Vanilla PPO, (2) PPO+FSP (History Pool), (3) PPO+Dense (with FSP and reward shaping), (4) Deep Q-Networks (DQN), (5) REINFORCE Manual and (6) REINFORCE Tianshou.

6.1 Win Rate Evaluation

Each agent's absolute strength was evaluated against a baseline agent which selects a legal move uniformly at random. The tournament win rates are detailed in Table 1.

Table 1: Performance of all trained agents against the Random Baseline (win % over 1000 games)

Agent	vs Random (Win %)
PPO+Dense	95.0%
Vanilla PPO	93.5%
REINFORCE Manual	88.5%
PPO+FSP	88.5%
REINFORCE TS	82.5%
DQN	62.0%

Observation The Policy Gradient methods (PPO variants and REINFORCE) all achieved a robust level of play (88.5% to 95% win rates). The DQN agent, despite significant sample time, only achieved 62% and was the weakest overall.

6.2 Comparative Tournament Analysis

While performance against a random agent establishes a baseline competency, the pairwise tournament reveals the structural hierarchy of the learned policies. The head-to-head win rates are summarized in the discussion below.

Dominance of Policy Gradient over Value-Based Methods A clear trend emerged where Policy Gradient methods (PPO and REINFORCE) significantly outperformed the Value-Based method (DQN). The DQN agent struggled against competent opponents, securing only a 3% win rate against Vanilla PPO and a 0% win rate against the Manual REINFORCE agent. Even against the Fictitious Self-Play (PPO Pool) agent, DQN only managed a 14% win rate. This suggests that in a pure self-play setting without tree search or extensive extensions (like Prioritized Experience Replay), policy optimization converges more reliably to a robust strategy than Q-learning for this specific state space.

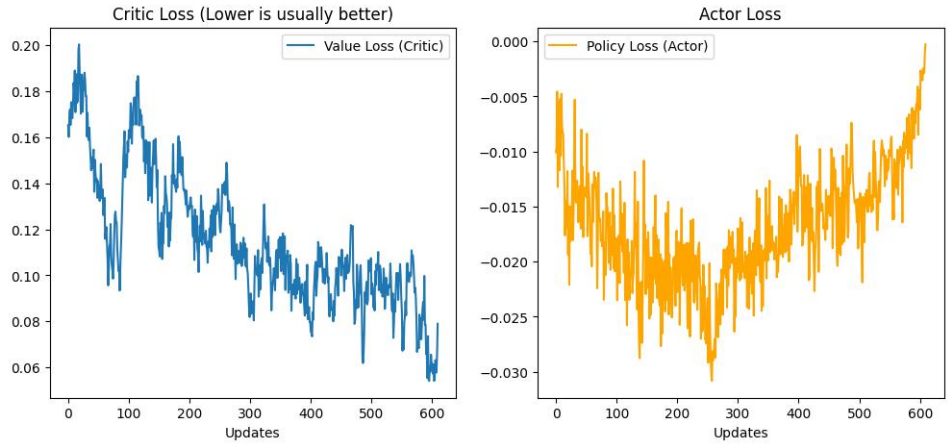
Impact of Reward Shaping (PPO Dense) The PPO agent trained with dense rewards (PPO Dense) demonstrated high consistency against weaker or unstable agents. It achieved a 100% win rate against the REINFORCE TS agent and a 92% win rate against DQN. However, when matched against the Vanilla PPO and Manual REINFORCE agents, the results were essentially even (hovering near 47–50% win rates). This indicates that while dense rewards accelerate learning and prevent collapse against weak opponents, they do not necessarily yield a higher ceiling than sparse-reward training given sufficient time.

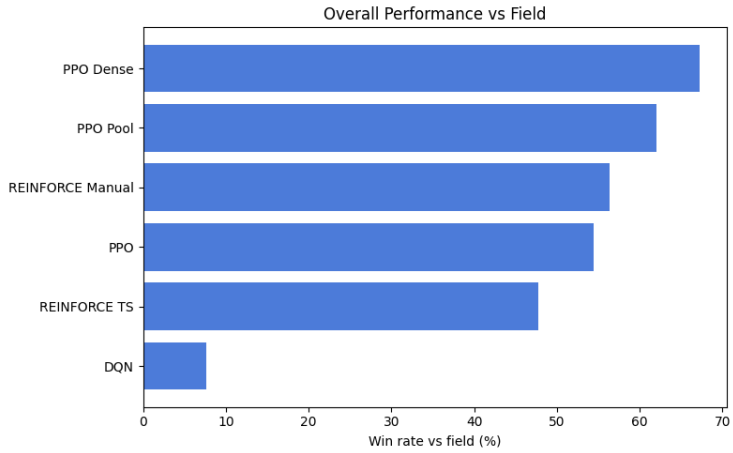
The Stability of REINFORCE Surprisingly, our manual implementation of REINFORCE performed on par with the more complex PPO algorithms, achieving roughly even splits (45–50% win rates) against the PPO variants. This contradicts some literature suggesting REINFORCE is too unstable for complex games; however, it highlights that with careful hyperparameter tuning and terminal credit assignment, simple policy gradients remain competitive in Connect Four.

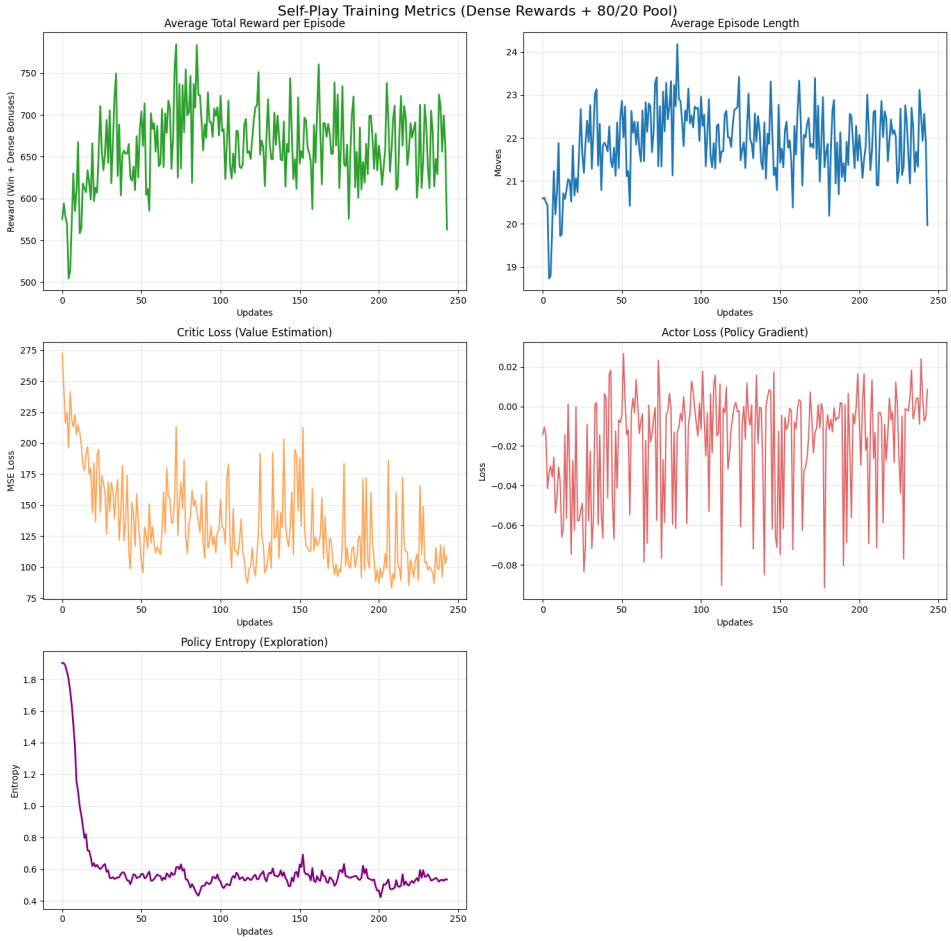
First-Player Advantage and Parity The tournament data highlighted the strong First-Player Advantage inherent in Connect Four. For example, in the matchup between REINFORCE TS and Vanilla PPO, the PPO agent won 100% of games, while in the reverse matchup, REINFORCE TS managed to win 85% of its games as Player 1. The strongest agents (PPO Dense and Manual REINFORCE) were distinguished by their ability to defend successfully as Player 2, effectively neutralizing the opponent's FPA.

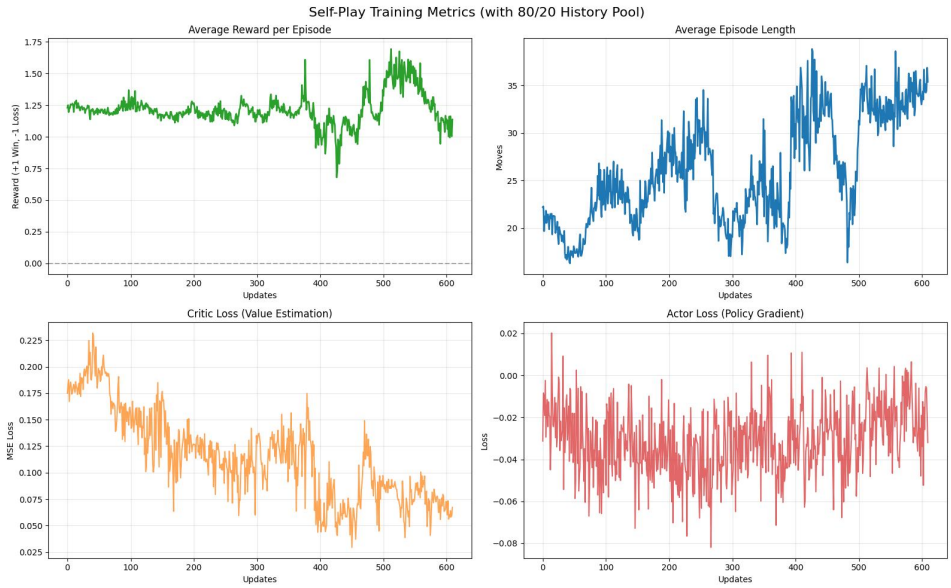
6.3 Figures

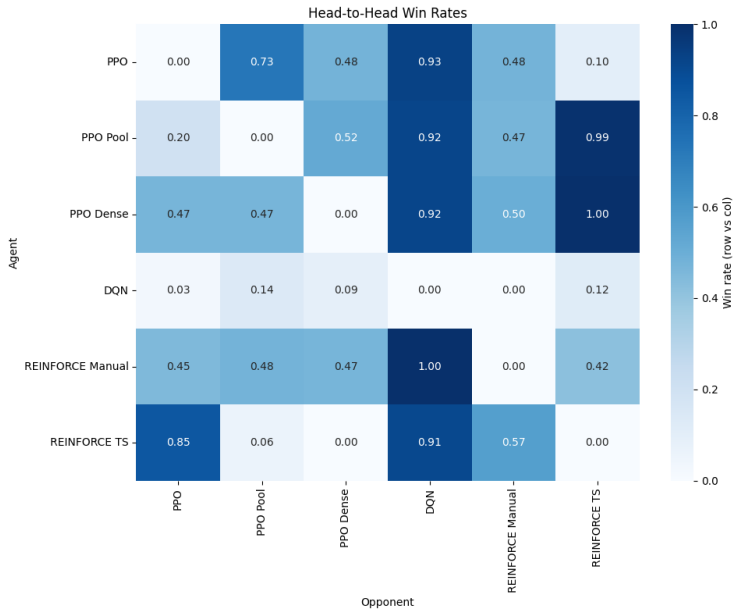
To support these findings, we include below the following figures:






Self-Play Training Metrics (Dense Rewards + 80/20 Pool)
Average Total Reward per Episode
Reward (Win + Dense Bonuses)
Updates
Average Episode Length
Moves
Updates
Critic Loss (Value Estimation)
MSE Loss
Updates
Actor Loss (Policy Gradient)
Loss
Updates
Policy Entropy (Exploration)
Entropy
Updates


Self-Play Training Metrics (with 80/20 History Pool)
Average Reward per Episode
Reward (+1 Win, -1 Loss)
Updates
Average Episode Length
Moves
Updates
Critic Loss (Value Estimation)
MSE Loss
Updates
Actor Loss (Policy Gradient)
Loss
Updates



7 Conclusion

This work demonstrates that model-free Reinforcement Learning, particularly Policy Gradient methods, can master Connect Four to a high degree of competency without the aid of Monte Carlo Tree Search. By modeling the game as an Egocentric MDP and utilizing Fictitious Self-Play, we successfully mitigated the reactive bias observed in naive self-play.

Our extensive tournament evaluation revealed that PPO and REINFORCE architectures significantly outperform standard Deep Q-Networks in this domain when training resources are equal. While reward shaping (PPO Dense) proved effective for consistency against weaker opponents, Vanilla PPO and Manual REINFORCE achieved comparable strategic ceilings, often resulting in draws or even splits against the strongest variants.

These findings suggest that for deterministic, perfect-information games of moderate complexity, explicit search is not strictly required to defeat intermediate baselines. However, the non-transitive dynamics observed in the tournament—where specific policies dominate others—highlight the continued necessity of population-based training (FSP) to ensure generalization. Future work will extend this approach to more complex variants, such as 7×8 grids or Connect-5, to test the scalability of search-free policy learning.

Code and Resources

The complete source code for all experiments, environment setup, model implementations, training pipelines, and evaluation scripts is available on GitHub:

`https://github.com/UmerSR/Connect-Four-RL`

The repository includes setup instructions, pretrained models, and logs to facilitate reproducibility of the results reported in this paper.

Author Contributions

The primary contributions to this work are divided as follows:

Umer Raja Led the foundational codebase structure, custom environment setup, and the formulation of the rewards mechanism. Responsible for the implementation and training of the Deep

Q-Networks (DQN) and REINFORCE agents. Authored core sections of the report and conducted the cross-agent meta-analysis and interactive GUI development.

Haris Rathore Led the Proximal Policy Optimization (PPO) implementation. Formulated the mathematical framework for the Egocentric MDP and the Zero-Sum GAE modification. Responsible for generating "versus random" evaluation metric. Authored core sections of the report.

Acknowledgments and Disclosure of Funding

We acknowledge the use of the Gymnasium library for environment simulation and PyTorch for neural network implementation. This project was conducted as part of the CS 6314 course requirements.

References

1. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., & Hassabis, D. (2018). *A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play*. Science, 362(6419), 1140–1144.
2. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal Policy Optimization Algorithms*. arXiv preprint arXiv:1707.06347.
3. Heinrich, J., & Lanctot, M. (2016). *Deep Reinforcement Learning from Self-Play in Imperfect-Information Games*. arXiv preprint arXiv:1603.01121.
4. Edelkamp, S., & Kissmann, P. (2008). *Symbolic classification of general two-player games*. In KI 2008: Advances in Artificial Intelligence, pp. 185–192.
5. Allen, J. D. (1988). *Solving Connect Four using Knowledge-Based Methods*. Master's thesis, University of Alberta.
6. Allis, L. V. (1988). *A Knowledge-Based Approach to Connect Four: The Game Is Solved*. Master's thesis, Vrije Universiteit Amsterdam.
7. Tesauro, G. (1995). *Temporal difference learning and TD-Gammon*. Communications of the ACM, 38(3), 58–68.
8. 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. (2015). *Human-level control through deep reinforcement learning*. Nature, 518(7540), 529–533.
9. Schmid, U., Maier, M., & Lang, M. (2019). *Benchmarking Deep Reinforcement Learning for Connect Four*. arXiv preprint arXiv:1906.00210.
10. Scheiermann, A., & Konen, W. (2022). *Comparing AlphaZero and Traditional Search for Board Games*. IEEE Transactions on Games, 14(3), 321–334.
11. Taylor, J., & Stella, F. (2024). *A Comparative Study of Reinforcement Learning and Search-Based Methods for Connect Four*. arXiv preprint arXiv:2405.16595.
12. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.