Extended Abstract

Motivation Hidden-information games that also demand spatial, long-horizon tactics—such as the board-and-card game *Sequence*—occupy an underexplored middle ground between classic perfect-information benchmarks (Go, Chess, Othello) and purely informational settings (Poker, Hanabi). Although *Sequence* is simple to learn and popular worldwide, the research community has lacked an open simulator, baseline agents, or a reproducible learning study. As a result, we lack a clear understanding of how much value explicit tree search contributes over modern policy–gradient methods when both are constrained to modest compute budgets under partial observability. By providing the first open benchmark and full code release, we bridge this gap and enable systematic comparison for future research in reinforcement learning and planning.

Method We introduce `SequenceEnv`, a Gym-compatible simulator that provides (i) a three-plane 10×10 board tensor (empty, self, opponent), (ii) a 96-bit one-hot hand vector, and (iii) rank counts for implicit card-tracking; actions factor into a 7-way *card* choice and a 100-way *square* choice, yielding 700 legality-masked logits. In addition to *Random* and *Greedy* baselines, we train two contrasting learners within this interface. The first, a recurrent PPO agent, embeds the board with a four-layer CNN, embeds the hand with a linear layer, fuses both in a 128-unit LSTM, and outputs factored softmax heads; it optimizes the clipped ratio objective with GAE, a decaying entropy bonus, and two shaped rewards that promote line extension and open-four blocking. The second, an AlphaZero-style MCTS agent, uses a 64-channel stem plus five residual blocks to supply priors and values to a PUCT search that, on every move, explores eight *root determinizations*—complete, history-consistent deals of the hidden deck—running 200 simulations per root and evaluating new leaves with an equal blend of network value and eight-turn random rollout. Both agents learn purely from self-play: PPO alternates 1024-step actor rollouts with five SGD epochs per batch, whereas MCTS performs supervised policy–value regression after every 2500 self-play games.

Implementation Experiments run on a single AWS `g4dn.xlarge` (T4 GPU, 4 vCPU, 16GB RAM). Sixteen CPU actors stream 1024-step trajectories to the PPO learner; 2.3M frames process in roughly 2 hours. MCTS expands 1600 nodes per move, and 20 2500-game blocks followed by one retraining epoch take around 6 hours.

Results Across a 10000-game all-play-all tournament, both learners dominate the Random and Greedy baselines, yet differ markedly in strength: PPO attains 75% win rate vs. Greedy, whereas MCTS achieves 85% and defeats PPO in 64.98% of direct match-ups. MCTS also finishes games 15% faster on average, reflecting more decisive tactical plans. A heatmap comparison of 1000 paired games reveals that PPO's placements concentrate along a few preferred lanes, whereas MCTS spreads its chips broadly across the board—demonstrating that explicit search not only wins more often but also explores richer spatial strategies. A small human pilot (26 games, 3 volunteers) mirrors self-play findings: humans win 13/26 games against PPO but manage only 8/26 versus MCTS, indicating the search agent already surpasses competent leisure players.

Discussion Our results confirm that search-augmented reasoning retains f a substantial edge under partial observability even when belief is approximated by simple determinization. The cost is higher inference latency and GPU saturation at eight determinizations and 200 simulations per. Limitations include implicit opponent modeling, removal of wildcard Jacks for simplicity, and the absence of more dense heuristics for reward shaping. Future extensions could integrate neural belief networks for more principled hidden-state estimation, value-guided rollout pruning, distributed search, and full rule variants to promote agents with additional tactical awareness.

Conclusion This work establishes `SequenceEnv`, the first open, end-to-end benchmark for *Sequence*, along with two complementary learning agents and baseline heuristics. Our experiments demonstrate that, even with identical input representations and comparable training budgets, explicit search via MCTS delivers both stronger play and more efficient game closures than a recurrent PPO policy. At the same time, the PPO agent offers fast inference and minimal compute requirements. By highlighting how learned policies and explicit search each excel under partial observability, we hope this benchmark will spark further research into games and decision problems that blend rich spatial observation with hidden information.

From Rules to Strategy: Teaching Reinforcement Learning Agents to Play *Sequence*

Nick Monozon
Institute for Computational and Mathematical Engineering (ICME)
Stanford University
nmonozon@stanford.edu

Abstract

We present a reproducible benchmark for the board game *Sequence*, which blends long-term spatial planning with hidden-hand inference and has, until now, lacked public simulators or learning baselines. Our contribution comprises (i) `SequenceEnv`, a Gym-compatible environment that captures the full game rules and supports fast agent prototyping; (ii) two heuristic opponents (Random and Greedy) that provide performance lower bounds; and (iii) two learning agents representing full range of planning approaches. A recurrent proximal policy optimization (PPO) agent, equipped with a CNN–LSTM encoder and a factored action head, reaches a 75% win rate against the Greedy baseline after 2.5M self-play frames. An AlphaZero-style Monte Carlo tree search (MCTS) agent, augmented with root determinization and a small residual policy–value network, attains an 85% win rate against Greedy and wins 65% of direct matches versus PPO, while completing games about 15% faster. Pairwise tournaments, ablation studies, and 26 human test games confirm that explicit search retains a decisive edge over pure policy gradients in this partially observable setting. All code and a human-play web interface are publicly released to facilitate further research on hybrid search, belief modeling, and full rule variants of *Sequence*.

1 Introduction

Learning to act under uncertainty is a large area of active research in reinforcement learning (RL) (Kaelbling et al., 1998; Hamrick et al., 2021). In game-playing domains with partial observability and adversarial opponents, agents must balance exploration of hidden information with exploitation of learned strategies, adapting in real time as new cards are revealed and countermoves are observed. Recent breakthroughs such as AlphaGo (Silver et al., 2017b), AlphaZero (Silver et al., 2017a), and Libratus (Brown and Sandholm, 2019) demonstrate two complementary approaches: deep look-ahead search on fully observed boards and strategy synthesis for hidden-information games.

Many popular games lie between these extremes, blending spatial planning with belief tracking (Heinrich and Silver, 2016; Perolat et al., 2022). The commercial board game *Sequence* highlights this middle ground: players alternate placing chips on a 10×10 grid to complete two non-overlapping 5-in-a-row "sequences." The first player to create two such sequences wins. With no knowledge of the opponent's hidden 7-card hand, players must balance immediate chip placement against longer-term board control and blocking opponent sequence formation.

Despite its pedagogical appeal, *Sequence* lacks an open simulator, baseline agents, or prior RL studies. To fill this gap, we contribute an efficient OpenAI Gym-compatible environment and two competitive learning agents, one based on recurrent proximal policy optimization (PPO) and the other on AlphaZero-style Monte Carlo tree search (MCTS) (Silver et al., 2017a; Cowling et al., 2012). We seek to analyze their behavior, strategy, and performance.

Specifically, we investigate whether explicit tree search maintains its edge when hidden information is handled via root determinization (Silver et al., 2017a), examine how the spatial placement patterns of policy–gradient and search-based agents diverge, and explore which training regimes can be executed effectively on modest hardware. We answer these questions through large-scale self-play, targeted ablations, and human trials.

2 Related Work

Early game-playing breakthroughs began with TD-Gammon, whose model-free temporal difference learning approach reached near expert-level Backgammon after 1.5 million self-play training games (Tesauro, 1995). Deep search then took center stage: AlphaGo and AlphaZero combined Monte Carlo Tree Search (MCTS) with deep policy-value networks to master perfect-information games such as Go, Chess, and Shogi (Silver et al., 2017a,b). In imperfect-information settings, counterfactual regret minimization (CFR) and its neural variants achieved superhuman performance in poker—most notably Libratus and DeepStack in heads-up no-limit Texas Hold'em (Brown and Sandholm, 2019; Moravčík, 2017).

Recent work has shifted toward reinforcement learning pipelines. Neural Fictitious Self-Play blends deep Q-learning with strategy averaging (Heinrich and Silver, 2016), while R2D3 introduced recurrent distributed double DQN for partially observable Atari tasks (Ozeki, 2018). On the policy–gradient side, recurrent PPO has been effective in hidden-state environments ranging from StarCraft II (Vinyals et al., 2019) to the cooperative card game Hanabi (Bard et al., 2020). Industry teams have also applied AlphaZero-style pipelines to hybrid games like Hearthstone and Magic: The Gathering, though details remain largely proprietary.

Open RL testbeds tend to sit at opposite ends of the observability spectrum: game-playing research focusing on fully visible boards with rich spatial structure (Go, Chess) or hidden-hand games with almost no board geometry (Poker, Hanabi) dominate the current literature. Few environments demand both long-horizon board planning and belief tracking. Moreover, the strongest imperfect-information agents—e.g., DeepStack or AlphaStar—typically require large compute budgets, so it remains unclear how well lighter training pipelines perform (Moravčík, 2017). By developing a Gym-compatible environment for *Sequence*, creating rule-based baselines, and implementing two agents that can be trained with modest compute budgets, we deliver a practical benchmark for this middle ground—one that lets us directly compare the strengths of policy–gradient and search-based methods under partial observability and spatial planning complexity.

3 Method

We present a complete end-to-end pipeline that includes (i) a Gym-compatible environment (`SequenceEnv`), (ii) two simple rule-based baselines that serve as performance references and opponents during training, and (iii) two learning agents: one based on recurrent Proximal Policy Optimization (PPO) and the other on AlphaZero-style Monte Carlo tree search.

3.1 Environment Design

To facilitate quick experimentation and reproducible benchmarking, we implement `SequenceEnv`, a Gym-compatible environment that includes all game rules, state representations, and legality checks.

Board layout The public board is a fixed 10×10 grid whose entries are drawn from the union of two standard 52-card decks (minus Jacks); the four corners are wild.

Observation At each turn t, each agent receives a three–component observation dictionary:

$$\mathcal{O}_t = \{\texttt{board} \in \{0,1,2\}^{10\times 10},\ \texttt{hand} \in \{0,1\}^{96},\ \texttt{counts} \in \mathbb{N}^{13}\}\,. \tag{1}$$

Here `board` indicates empty/self/opponent chips, `hand` is a one–hot encoding of the 7 cards currently held, and `counts` tallies visible cards for each rank, enabling implicit belief updates.

Action representation Legal moves are encoded as a two-factor discrete choice (c, p) where $c \in \{1, \ldots, 7\}$ selects one of the 7 cards in the agent's hand and $p \in \{1, \ldots, 100\}$ selects a board square. Because our implementation of *Sequence* removes all Jacks from the double deck, every remaining card face appears on exactly one or two squares. A move (c, p) is legal if and only if the chosen hand card matches the face printed on square p and that square is currently empty. At each step the environment supplies a 700-bit legality mask that zeroes out the logits of invalid (c, p) pairs before softmax.

Reward structure Terminating wins (two non-overlapping 5-in-a-row "sequences") grant $+1$, losses -1, and draws 0. To accelerate early exploration we add two shaped components: (i) *line extension* $+0.05\Delta L$ where L is the longest contiguous chip-run after the agent's move, and (ii) *open–four blocking* $+0.1$ whenever the agent blocks an opponent from extending a 4-in-a-row to a full sequence. These shaped rewards are applied only during PPO training, while MCTS self-play relies solely on sparse terminal rewards $(+1, 0, -1)$.

3.2 Baseline Agents

To evaluate our learning-based agents, we implement two simple, rule-based agents. These agents provide performance lower bounds and help quantify the strategic depth achieved by our PPO and MCTS approaches.

1. The Random agent queries the environment for all legal moves at each turn and selects one uniformly at random. It guarantees unbiased coverage of the action space but has no awareness of board tactics.
2. The Greedy agent evaluates every legal move by estimating its impact on contiguous chip-runs of lengths 2–5 according to the score given by

$$\text{score} = \sum_{k=2}^{5} w_k \times (\#\text{ of } k\text{-runs}) \tag{2}$$

with $(w_2, w_3, w_4, w_5) = (1, 5, 10, 100)$. The agent then chooses the legal move that maximizes this score. This heuristic rapidly builds medium-length chip-runs, but lacks lookahead and blocking ability, so it can be outmaneuvered by deeper tactics.

3.3 Recurrent PPO Agent

Our proximal policy optimization (PPO) agent is built for partial observability and a factored action space. Three binary chip planes (empty, self, opponent) form the spatial core. A static card-ID map supplies an additional feature plane that tells the network which face occupies each square. The private 7-card hand is a 96-bit one-hot vector. Chip and card planes are embedded and processed by a shallow four-layer CNN; the hand vector passes through a linear layer. The resulting features are concatenated and fed into a 128-unit LSTM that maintains cross-move memory.

From the LSTM state we emit two softmax heads: a 7-way *card head* and a 100-way *square head*. Their Kronecker product yields the 700-dimensional policy, which is masked by the legality bit-vector before sampling and loss computation. This factorization exploits the structure of the game and accelerates learning compared to a flat softmax (Schulman et al., 2017).

For the critic, a single linear layer projects the LSTM state to a scalar value estimate $V_\theta(s)$. The policy and value networks are trained jointly by minimizing the composite PPO loss

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{clip}}(\theta) + \frac{1}{2}(V_\theta(s_t) - R_t)^2 + c_e H(\pi_\theta(\cdot|s_t))\,, \tag{3}$$

where $\mathcal{L}_{\text{clip}}$ is the standard clipped ratio policy loss that stabilizes updates, R_t is the GAE target combining sparse terminal reward with the two shaped bonuses, and $H(\pi)$ is the policy entropy, weighted by a coefficient c_e that decays during training to phase out forced exploration (Schulman et al., 2016). Thus the critic's value term smooths the sparse game outcome with intermediate tactical feedback, while the entropy term keeps the policy exploratory early on. The clipped actor term learns to maximize these shaped returns without letting the policy drift too far in a single step.

During the first 400 PPO training updates, the agent faces the Greedy and Random agents 50% of the time each, supplying diverse yet tractable opposition while the policy stabilizes. After, opponents are drawn on a per-game basis with probabilities 70% from a rolling pool of the five most recent PPO checkpoints (self-play), 15% Greedy, and 15% Random. This schedule maintains behavioral variety and steadily increases difficulty. Each iteration alternates one batch of data collection with several epochs of mini-batch SGD updates (Schulman et al., 2018).

3.4 AlphaZero–style MCTS Agent

Our AlphaZero-style agent pairs a lightweight residual CNN with MCTS, handling hidden information through root determinization. The policy–value network ingests 99 binary planes: three chip layers (empty, self, opponent) and one plane for each of the 96 card identities in the current hand. A 3×3 stem convolution with 64 channels feeds five residual blocks; two task heads then branch off. The policy head ends in 700 logits (the Kronecker product of 7 hand indices and 100 board indices) while the value head applies a final $\tanh$ activation to produce a scalar in $[-1, 1]$. For this scalar, $+1$ corresponds to a certain win, -1 a certain loss, and 0 a draw.

At decision time, the agent samples K complete "deal" completions that fill the opponent's hand and the undealt deck consistently with public history. On each determinization it runs S PUCT simulations, selecting children according to

$$U(s,a) = Q(s,a) + C_{\text{PUCT}}\, P(s,a) \frac{\sqrt{N(s)}}{1 + N(s,a)}\,, \quad C_{\text{PUCT}} = 1.5\,. \tag{4}$$

Here, $Q(s,a)$ is the running mean Monte Carlo value of edge (s,a), $P(s,a)$ is the prior probability supplied by the policy head, $N(s)$ and $N(s,a)$ are, respectively, the visit counts of the parent node and the child edge, and C_{PUCT} controls the exploration–exploitation tradeoff, set according to (Czech et al., 2020). During each move the agent samples K determinizations, runs S simulations on each, and evaluates newly expanded leaves with an equal-weight combination of the network value and an eight-turn random rollout score:

$$V_{\text{leaf}} = 0.5 v_\theta(s_{\text{leaf}}) + 0.5 \hat{z}_{\text{rollout}}\,. \tag{5}$$

After all simulations finish, visit counts from the K roots are averaged to form a search policy π. For the first 20 turns of every game the action is sampled from π (temperature $\tau = 1$) to encourage exploration; thereafter the arg-max is taken as $\tau \to 0$.

Each self-play game yields a sequence of triples (s_t, π_t, z), where s_t is the encoded state, π_t the visit-count policy just computed, and $z \in \{-1, 0, 1\}$ the final outcome from the current player's perspective. After every block of 2500 games we shuffle these records and run a single epoch of stochastic gradient descent (Adam, learning rate `1e-3`, batch size `1024`). The objective combines a cross-entropy term that matches the network policy to π_t, an MSE term that regresses the value head onto z, and ℓ_2 regularization (`1e-4`). To stabilize search, we keep a lagged copy of the parameters for tree expansion and update it toward the latest weights with Polyak averaging ($\tau_{\text{polyak}} = 0.005$) after each training step (Cowling et al., 2012).

4 Experimental Setup

Task and data generation Our objective is to train two agents that win two-player *Sequence* by creating two non-overlapping 5-in-a-row sequences before their opponent does. All training and evaluation data is generated on-the-fly through self-play; no offline datasets are used. For the recurrent PPO agent, we track learning progress by the total number of environment frames processed. For the MCTS agent, we report performance based on the number of completed self-play games simulated against a rolling pool of previous MCTS checkpoints.

Evaluation metrics Agent performance is evaluated in four ways. First, *win rate vs. baselines* quantifies raw strength by measuring victories against the Greedy and Random heuristic agents. Second, *pairwise win rates* report results from a large-scale all-play-all tournament among Random, Greedy, PPO, and MCTS, illustrating relative ranking under identical conditions. Third, *win rate vs. humans* evaluates transfer to real opponents using results from a 26-game pilot study. Lastly,

Rolling Elo tracks each checkpoint's skill relative to its own training history to determine if the agent is learning better strategy: every new policy enters a tournament of past snapshots, yielding an Elo curve over training episodes (Silver et al., 2017a).

Hardware All runs were executed on an AWS `g4dn.xlarge` instance equipped with a 16GB NVIDIA T4 GPU, 4 vCPUs, and 16GB RAM.

Recurrent PPO training Sixteen CPU actors generate batches of `1024` steps that stream asynchronously to the GPU. Each batch triggers five epochs of mini-batch SGD (size `2048`) using Adam (learning rate `3e-4`, weight decay `1e-5`). We use $\gamma = 0.99$ and $\lambda = 0.95$ for GAE and fix the clip range at $\varepsilon = 0.2$, the optimal value in our ablation study (Table 1). The entropy bonus decays linearly from 0.01 to 0.001 over the first 40% of updates. Training consumes 2.3M environment frames, which takes around 2 hours.

MCTS training The search agent employs $K = 8$ root determinizations and $S = 200$ PUCT simulations per determinization, with constant $C_{\text{PUCT}} = 1.5$ to balance exploration and exploitation, expanding 1600 nodes in total (Czech et al., 2020). Self-play proceeds in blocks of 2500 games; the policy–value network is then trained for a single epoch on the resulting (s_t, π_t, z) triples using Adam (learning rate `1e-3` and batch size `1024`). This takes roughly 6 hours.

Ablation procedure To evaluate hyperparameter sensitivity, we vary the PPO clip range $\varepsilon \in \{0.10, 0.15, \ldots, 0.30\}$ and the MCTS simulation budget $S \in \{50, 100, 200, 400\}$. Each configuration is trained with five random seeds; results are summarized in Table 1.

The clip range ε sets how far the new policy is allowed to deviate per update from the behavior policy that generated the data. A small ε enforces conservative steps, leading to stable but slow learning, while a large ε lets the policy move aggressively, risking divergence. Because our game has sparse rewards and partial observability, finding the optimal ε for stability and policy improvement speed is critical. Across the range of candidates for ε, we track the final Elo and the number of environment frames (in millions) required to surpass the 70% Greedy win rate threshold.

For the MCTS agent, strength depends on how many simulations explore the tree and how strongly priors steer the search. Raising S improves search accuracy but increases latency and training cost; lowering S does the opposite. In parallel, the exploration constant $C_{\text{PUCT}} = 1.5$ balances prior-guided exploration vs. value exploitation. Varying S while fixing $C_{\text{PUCT}} = 1.5$ reveals the performance vs. compute tradeoff and highlights the point at which additional simulations yield diminishing Elo gains. Across the range of candidates for S, we track the final Elo, per-move inference latency, and total nodes expanded.

Table 1: Ablation results (mean $\pm$ s.e. over five seeds). PPO: final Elo and frames (in millions) needed to first exceed 70% win rate vs. Greedy. MCTS: final Elo, per-move inference latency, and total nodes expanded.

PPO — clip range ε			**MCTS — simulations S**			
ε	Elo	>70% (M)	S	Elo	Time (ms)	Nodes
0.10	1420 ± 7	2.1	50	1470 ± 5	211	400
0.15	1475 ± 6	1.7	100	1570 ± 4	383	800
0.20	$\mathbf{1510 \pm 4}$	**1.4**	**200**	$\mathbf{1650 \pm 3}$	**900**	**1600**
0.25	1505 ± 5	1.5	400	1665 ± 2	1750	3200
0.30	1500 ± 6	1.9	—	—	—	—

From the ablation results, we select $\epsilon = 0.20$ for PPO as it best balances update stability and learning speed, and we choose $S = 200$ for MCTS since larger simulation budgets yield only marginal Elo gains at much higher computational cost. These configurations are adopted for all subsequent experiments.

5 Results

5.1 Quantitative Evaluation

5.1.1 Learning Curves

Figure 1 follows the PPO agent for 1000 steps, or about 2.5M environment frames. The left panel shows a steady climb in win rate versus the Greedy baseline: starting below 10%, the curve breaks the 50% threshold around update 200 and plateaus near 75% by update 800. The right panel plots the average reward per step and has a similar trend: from a slightly negative baseline it climbs to 0.5 within the first 300 updates and then continues increasing more slowly to 0.6 by update 600 before plateauing. PPO appears to learn to recognize local tactical cues (reward spike) and only later converts them into consistent wins, which is a trajectory typical of reward sparsity.

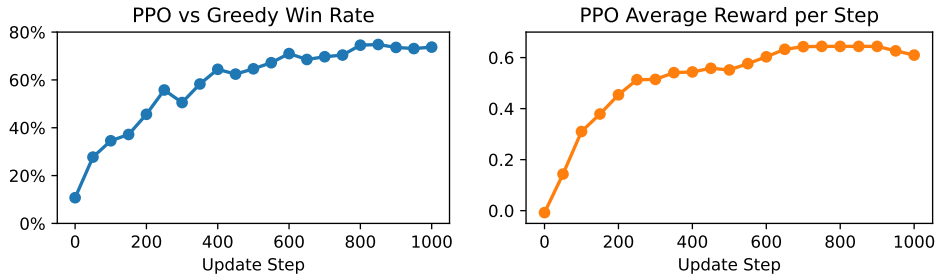

Figure 1: PPO training: win rate vs. Greedy (left) and shaped reward per step (right).

Figure 2 follows the MCTS for retraining after every 2500 games. After an early exploration spike, the MCTS win rate against Greedy increases to >80% by update 150 and then oscillates between roughly 75% and 85% for the remainder of training. These oscillations are an artefact of our batched self-play regime: each 2500-game block is played with a frozen network; retraining the policy–value model inserts a fresh set of priors, momentarily upsetting the tree statistics and nudging the win rate up or down until the search settles. Despite the oscillations, the win rate never drops below 75% and its upper envelope creeps higher with each iteration. The Elo curve, which is measured against a rolling pool of previous snapshots, climbs more smoothly, rising consistently from about 1,000 to 1,650 before slowing down, which mirrors the diminishing returns pattern seen in our ablation study in Table 1.

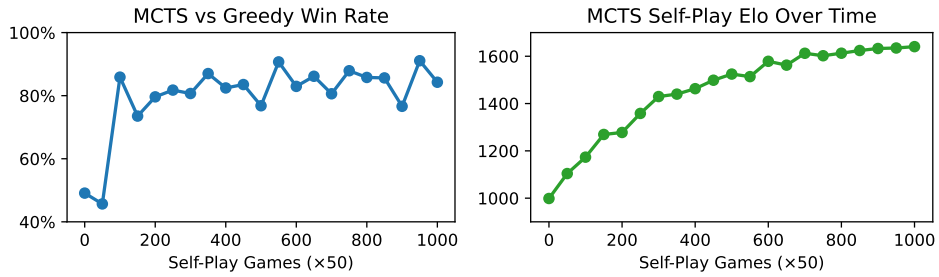

Figure 2: MCTS self-play: win rate vs. Greedy (left) and Elo vs. rolling snapshot pool (right), where snapshots are taken every 2.5k updates.

Taken together, the two plots underscore complementary learning dynamics: PPO progresses steadily but hits a ceiling, whereas MCTS advances in punctuated jumps, wobbling after every network refresh yet ultimately converging to the higher skill plateau.

5.1.2 Pairwise Win Rates

To compare agents under identical conditions we ran a large-scale all-play-all tournament in which every pair of agents played 10,000 games. With four agents (Random, Greedy, PPO, MCTS) this

yields the 6 match-ups shown in Table 2, totaling 60000 games. Each entry corresponds to the percent of the time that the row agent beat the column agent. To eliminate any first-mover bias, starting positions alternated so that each agent moved first in exactly half of its games. Since some games ended in draws, the pairwise win rates may not sum to 100%

Table 2: Pairwise win rates

	Random	Greedy	PPO	MCTS
Random	—	16.13%	10.03%	9.63%
Greedy	**83.73%**	—	23.42%	14.73%
PPO	**89.75%**	**76.37%**	—	33.95%
MCTS	**90.31%**	**85.14%**	**64.98%**	—

The aggregate statistics reinforce the Elo ordering that emerges during training. Both learning agents greatly outperform the baselines: PPO wins 76.37% of its games against Greedy and almost 90% against Random, while MCTS is stronger still, achieving win rates of 85.14% and 90.31% respectively. The direct match-up between PPO and MCTS is most revealing: MCTS beats PPO in 64.98% of their 10,000 games, confirming that explicit tree search confers a substantial strategic advantage even under partial observability.

5.1.3 Human–vs–agent games

We also gathered a small set of human vs. agent games (Table 3). Three volunteer players of various skill levels completed 26 games in total—7, 8, and 11 games, respectively—against each agent. In aggregate, human win rates were 50% against PPO but only 31% against MCTS. No single player managed to exceed a 36% success rate versus MCTS, suggesting that even casual human competitors struggle against search-augmented reasoning. By contrast, the 50% split against PPO indicates that the learned policy sits squarely at the level of an average leisure-time player: it can hold its own but is easily outmaneuvered by a moderately skilled human. Although these pilot results are limited by sample size, they mirror the self-play gap and imply that MCTS not only excels in simulated benchmarks but also transfers its strategic advantage to real opponents.

Table 3: Human vs. agent outcomes. Values are *human wins / games* (win rate).

	Games	Wins vs. PPO	Wins vs. MCTS
Player A	7	3 / 7 (43%)	2 / 7 (29%)
Player B	8	4 / 8 (50%)	2 / 8 (25%)
Player C	11	6 / 11 (55%)	4 / 11 (36%)
Aggregate	26	13 / 26 (50%)	8 / 26 (31%)

5.2 Qualitative Analysis

5.2.1 Game Length

Beyond win rates and Elo, we also compared how quickly PPO and MCTS secured victories by plotting the distribution of winning game lengths (in turns) from our 10000 all-play-all matches against both Random and Greedy baselines (Figure 3).

Against Random, MCTS concludes games substantially faster than PPO. The MCTS distribution tightly concentrated around 55–65 turns (mean 60.58, median 61), whereas PPO games center around 65–75 turns (mean 70.42, median 71). This indicates that MCTS's explicit search rapidly identifies tactical sequences leading to victory, while PPO's learned policy engages in longer exploratory play before securing victory.

When facing the Greedy heuristic, both agents finish even more quickly, reflecting Greedy's predictable playing style. Here, MCTS median game length drops to 53 turns, and PPO to 61 turns. The sharper left shift of MCTS's curve shows that search can exploit Greedy's one-step scoring rule to end games more quickly. PPO still requires more turns to outmaneuver Greedy's heuristics, resulting in a broader spread of game lengths.

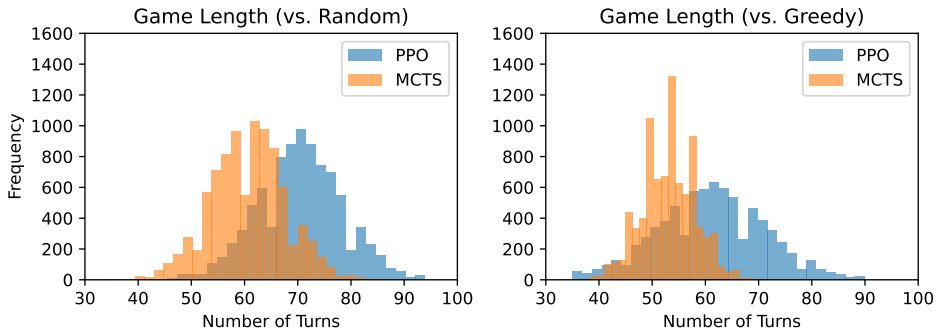

Figure 3: Histogram of turn counts for winning games only. Left: against Random. Right: against Greedy. In both, PPO observations are blue and MCTS observations are orange.

Moreover, PPO's histograms exhibit a heavier tail, indicating occasional protracted games where neither agent can form two sequences, whereas MCTS's curves remain relatively narrow, rarely exceeding 80 turns against Random and never exceeding 70 turns against Greedy. These distributions underscore that search-augmented reasoning not only achieves higher win rates than memory-based policies but also closes out games more efficiently.

5.2.2 Play Style Comparison

To analyze the play style of PPO and MCTS, we ran 1000 paired games against a "pseudorandom" opponent: within each pair, the opponent's moves are identical, but each game uses a new deck shuffle. This design guarantees that PPO and MCTS face exactly the same opponent behavior in every match-up, so any differences in outcome or board coverage reflect only their own decision policies. For each game we logged the win/loss result for the learner and every chip placement it made, enabling direct comparison of both aggregate win rates and spatial play patterns.

Each paired trial uses a different shuffle of the deck—every round is an independent draw—while the pseudorandom opponent always repeats the same move sequence within each comparison (when possible). Of these 1000 match-ups, PPO won 893, while MCTS won 992.

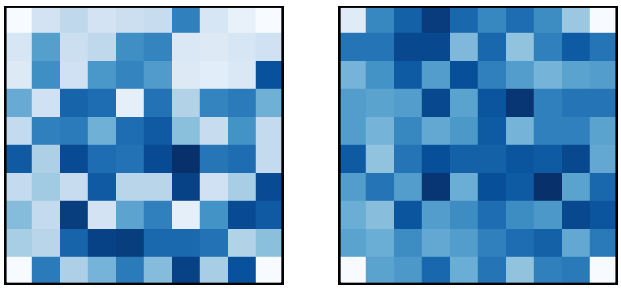

Figure 4: Aggregated relative frequency heatmaps of chip placements across 1000 games. Left: PPO agent. Right: MCTS agent. Darker shades of blue indicate more frequent chip placement.

PPO's placements (left panel) remain highly focal: two dense diagonal bands extend from the center, with especially dark streaks near the lower middle column and rows, while peripheral rows and extreme corners are rarely selected. This suggests the PPO policy discovers a small set of local templates early on and then reinforces them, forgoing exploration of other board regions.

By contrast, MCTS (right panel) displays comparatively more uniform coverage. Although squares in and around the center are still selected the most, very few squares remain unused. Peripheral rows and columns have moderate chip placement, indicating that look-ahead MCTS rollouts routinely uncover profitable moves beyond PPO's habitual choices. The overall darker shading also implies that MCTS explores more options per game before converging, yielding richer spatial coverage.

These contrasting patterns help explain the win rate differences in Section 5.1.2. PPO's narrow focus can produce rapid victories when its "preferred" lanes stay clear, but it struggles to adapt if those lanes are blocked. MCTS's broader exploration reduces this risk, while computing longer decision paths, resulting in a higher aggregate win rate against the same pseudorandom opponent.

6 Discussion

Our findings show that an AlphaZero-style MCTS agent with root determinization and a modest compute budget clearly outperforms a recurrent PPO policy in *Sequence*. This advantage, however, has three notable constraints.

First, the method is computationally expensive: eight determinizations and 200 PUCT simulations already saturate a T4 GPU. Scaling to richer card decks or deeper search will demand smarter pruning or distributed rollouts. Second, both agents reason about the opponent's hand only implicitly—PPO via its LSTM memory and MCTS by repeatedly resampling complete deal states—leaving untapped headroom for explicit Bayesian or neural belief models that could steer search toward higher probability lines of play (Moreno et al., 2021). Third, we simplified the rule set by removing wildcard Jacks and reduced shaped rewards down to two heuristics. While these choices speed benchmarking, they potentially ignore tactical nuances of the game. As such, reproducing the study under full rules is a natural extension to our work.

Even within these limits, the benchmark reveals sharp behavioral contrasts between agents: PPO quickly locks onto squares that it has learned to be "good" and ventures elsewhere much more rarely. In contrast, MCTS explores the entire board, producing quicker victories and a higher win rate against human and nonhuman opponents alike. Taken together, these results establish *Sequence* as a tractable yet challenging testbed for studying how search, memory, and explicit belief tracking interact in a partial observability environment.

7 Conclusion

We present the first open benchmark for *Sequence*—complete with a Gym-compatible environment, rule-based baselines, and two learning agents—and use it to compare a recurrent PPO policy with an AlphaZero-style MCTS agent. Across 10000-game tournaments, both agents far exceed baseline performance, but MCTS holds a clear edge over PPO, winning 65% of their head-to-head games. Play style analysis links this advantage to MCTS's broader board coverage, and a small human–vs–agent study (26 games) confirms its real-world strength: PPO wins 13/26 games, while MCTS wins 18/26.

Explicit tree search via root determinization continues to outperform pure policy–gradient methods, even under modest compute constraints. In future work, we will integrate neural belief modules to reduce determinization overhead, expand the policy–value network to enhance search quality, and reintroduce wildcard Jacks to evaluate full rule complexity. All code has been made publicly available; see Section A.

8 Team Contributions

- **Nick Monozon:** Implemented the Gym-compatible `SequenceEnv` with optimized rollouts; developed the recurrent PPO agent (LSTM actor–critic, shaped rewards, custom curriculum); built the AlphaZero-style MCTS pipeline (root determinization, residual policy–value network, mixed leaf evaluation); implemented distributed training, hyperparameter sweeps, and run logging; ran 60000 automated pairwise match-ups, ablation studies, and a 26-game human pilot study; performed all quantitative and qualitative analyses; built a Streamlit web app for human play; authored the final report.

Changes from Proposal Compared to the original proposal, we eliminated the two-stage mini-*Sequence* curriculum in favor of a single warm-up phase (400 PPO updates vs. Greedy/Random) followed by a mixed opponent schedule (70% self-play, 15% Greedy, 15% Random). We streamlined the action space by removing all Jacks, simplifying legality checks and masking. The shaped reward scheme was re-tuned to +0.05 per unit of line extension and +0.10 for blocking an open four, which

yielded faster and more stable early learning. We also added systematic ablation studies to validate our choice of PPO clip range and MCTS simulation budget.

References

Nolan Bard, Jakob N. Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H. Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, Iain Dunning, Shibl Mourad, Hugo Larochelle, Marc G. Bellemare, and Michael Bowling. 2020. The Hanabi challenge: A new frontier for AI research. *Artificial Intelligence* 280 (March 2020), 103216. https://doi.org/10.1016/j.artint.2019.103216

Noam Brown and Tuomas Sandholm. 2019. Superhuman AI for multiplayer poker. *Science* 365, 6456 (2019), 885–890. https://doi.org/10.1126/science.aay2400 arXiv:https://www.science.org/doi/pdf/10.1126/science.aay2400

Peter I Cowling, Edward J Powley, and Daniel Whitehouse. 2012. Information Set Monte Carlo Tree Search. *IEEE Transactions on Computational Intelligence and AI in Games* 4, 2 (2012), 120–143.

Johannes Czech, Patrick Korus, and Kristian Kersting. 2020. Monte-Carlo Graph Search for AlphaZero. arXiv:2012.11045 [cs.AI] https://arxiv.org/abs/2012.11045

Jessica B. Hamrick, Abram L. Friesen, Feryal Behbahani, Arthur Guez, Fabio Viola, Sims Witherspoon, Thomas Anthony, Lars Buesing, Petar Veličković, and Théophane Weber. 2021. On the role of planning in model-based deep reinforcement learning. arXiv:2011.04021 [cs.AI] https://arxiv.org/abs/2011.04021

Johannes Heinrich and David Silver. 2016. Deep Reinforcement Learning from Self-Play in Imperfect Information Games. In *ICML Workshop on Deep Reinforcement Learning*.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101, 1 (1998), 99–134.

Matej et al. Moravčík. 2017. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science* (2017).

Pol Moreno, Edward Hughes, Kevin R. McKee, Bernardo Avila Pires, and Théophane Weber. 2021. Neural Recursive Belief States in Multi-Agent Reinforcement Learning. arXiv:2102.02274 [cs.LG] https://arxiv.org/abs/2102.02274

Jeff et al. Ozeki. 2018. R2D3: Recurrent experience replay in distributed reinforcement learning. *arXiv preprint arXiv:1811.00260* (2018).

Julien Perolat, Bart De Vylder, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer, Paul Muller, Jerome T. Connor, Neil Burch, Thomas Anthony, Stephen McAleer, Romuald Elie, Sarah H. Cen, Zhe Wang, Audrunas Gruslys, Aleksandra Malysheva, Mina Khan, Sherjil Ozair, Finbarr Timbers, Toby Pohlen, Tom Eccles, Mark Rowland, Marc Lanctot, Jean-Baptiste Lespiau, Bilal Piot, Shayegan Omidshafiei, Edward Lockhart, Laurent Sifre, Nathalie Beauguerlange, Remi Munos, David Silver, Satinder Singh, Demis Hassabis, and Karl Tuyls. 2022. Mastering the game of Stratego with model-free multiagent reinforcement learning. *Science* 378, 6623 (Dec. 2022), 990–996. https://doi.org/10.1126/science.add4679

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2016. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *ICLR*.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2018. High-Dimensional Continuous Control Using Generalized Advantage Estimation. arXiv:1506.02438 [cs.LG] https://arxiv.org/abs/1506.02438

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347* (2017).

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. 2017a. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. arXiv:1712.01815 [cs.AI] https://arxiv.org/abs/1712.01815

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George Driessche, Thore Graepel, and Demis Hassabis. 2017b. Mastering the game of Go without human knowledge. *Nature* 550 (10 2017), 354–359. https://doi.org/10.1038/nature24270

Gerald Tesauro. 1995. Temporal difference learning and TD-Gammon. *Commun. ACM* 38, 3 (March 1995), 58–68. https://doi.org/10.1145/203330.203343

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 7782 (2019), 350–354. https://doi.org/10.1038/s41586-019-1724-z

A Code

All code developed for this project, including a human-play web app, is publicly available in the `sequence-agent` repository: `https://github.com/nickmonozon/sequence-agent`.