Reinforcement Learning Agent for the Mill Game

A Comparative Study of Training Against Multiple Baselines

Anonymized
Faculty of Mathematics, Natural Sciences
and Information Technologies,
University of Primorska, Slovenia

Anonymized
Faculty of Mathematics, Natural Sciences
and Information Technologies,
University of Primorska, Slovenia

***Abstract*—We develop a Deep Reinforcement Learning agent for Nine Men's Morris using Proximal Policy Optimization with action masking. Our primary contribution is a systematic comparison of training regimes: random opponents, minimax-based opponents, and self-play. Results show that RL agents achieve performance comparable to classical minimax approaches. Training against structured minimax opponents produces superior learning efficiency and final performance compared to both random-opponent training and self-play within the evaluated training duration. However, self-play demonstrates stronger continued improvement in later stages, suggesting it may eventually surpass fixed-opponent methods given sufficient computational resources. These findings indicate that opponent selection significantly impacts learning efficiency, with moderately challenging structured opponents providing the most effective training signal for resource-constrained settings.**

I. Introduction

Nine Men's Morris is a classic strategic board game characterized by three distinct phases of play. In this paper, we investigate optimal training configurations for a reinforcement learning–based agent capable of playing the game effectively. The task poses two principal challenges. First, the sparsity of legal moves renders standard reinforcement learning algorithms, such as DQN and PPO, largely impractical. Second, the presence of multiple game phases effectively transforms the problem into three interconnected yet fundamentally different subgames. To advance the development of an effective reinforcement learning agent for this game, we investigate a modified Proximal Policy Optimization approach and focus specifically on the impact of different training regimes.

A. Research Questions

This study seeks to answer the following questions:

1) Is RL agent comparable in performance to classical search-based Minimax agent?
2) How does the choice of training regime influence the learning efficiency and final performance of a reinforcement learning agent in *Nine Men's Morris*?

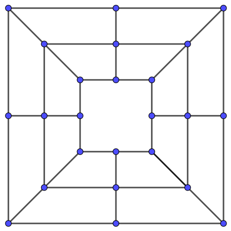
B. Related Work

The application of RL to Nine Men's Morris has its roots in classical heuristic search. Gasser famously solved the game in 1996, proving it to be a draw under perfect play using a combination of retrograde analysis and minimax search [1]. As part of prior work, we implemented a Minimax-based agent, exploring classical heuristic optimizations [2]. This implementation serves as a reference point and a baseline for evaluating the effectiveness of reinforcement learning approaches presented in this paper.

II. The problem domain

A. Nine Men's Morris Game Mechanics

Nine Men's Morris is a two-player, zero-sum, deterministic strategy board game played on a fixed graph, that in our version of the game, consists of three concentric squares each consisting of 8 vertices that are connected in the following way:



In our version of the game, each player controls nine pieces. Gameplay is divided into three distinct phases. During the *placement phase*, players alternately place their pieces on empty board intersections until all eighteen pieces have been deployed. Once placement is complete, the game transitions to the *movement phase*, in which players slide one of their pieces to an adjacent empty position along the board's connections. If a player is reduced to exactly three pieces, the game enters the *flying phase*, allowing that player to move a piece to any vacant position on the board regardless of adjacency. A *mill* is formed when three pieces belonging to the same player align along a valid line; forming a mill grants the player the right to remove one of the opponent's pieces. Pieces that are themselves part of a mill may not be captured unless no other opponent pieces are available for removal. The game terminates with a loss when a player, at the start of their turn, has fewer than three remaining pieces or is unable to make a legal move, and it is declared a draw if a total of 200 moves are played without a decisive outcome.

B. Simulation Overview

We took the existent implementation of experimental environment for the game that was created by Domen Šoberl [3], that is built on top of Gymnasium [4] and Pettingzoo [5] frameworks. It enforces all game rules and turn-based interactions, and since it complies to the standard Gymnasium API, it is easily integrated with RL frameworks.

C. Observation Space

The original environment's observation space was augmented as follows:

Normalized Board State (0–24)

- 1.0: Represents the agent's own piece.
- -1.0: Represents an opponent's piece.
- 0.0: Represents an empty tile.

Game State Features (25–30)

- Normalized piece counts for the agent and the opponent.
- Number of remaining pieces to be placed by the agent and the opponent.
- Current game phase for the agent.
- Total number of turns elapsed.

D. Action Space, Action Masking, Reward

The original action space is multi-discrete (selecting a source, a destination, and if applicable, a removal position). To represent it in the network output layer, we perform an integer encoding:

$$a = src \times 25^2 + dst \times 25 + remove \tag{1}$$

To handle the sparsity of legal actions we perform Action Masking (1 for legal, 0 for illegal).

We adopt the same reward structure as the original environment:

$$\begin{cases} +1 & \text{if the agent captures a piece,} \\ -1 & \text{if the agent loses a piece.} \end{cases} \tag{2}$$

Initially, we considered redefining the rewards to a classical win/lose signal with a small penalty for longer games. However, after some considerations, we concluded that the original reward formulation provides a well-balanced signal: Losing more pieces than captured - results in a negative reward, capturing more pieces than lost - results in a positive reward, naturally aligning with winning and losing conditions, while also increasing the reward for losing less and decreasing for losing more. Therefore, we chose to retain this reward system.

III. Methods

A. Choice of RL Algorithm

In designing the agent, we considered two prominent Deep Reinforcement Learning (DRL) architectures: **Deep Q-Networks (DQN)** [6] and **Proximal Policy Optimization (PPO)** [7]. Unlike classical RL, which relies on tabular representations of state-action values, Deep Reinforcement Learning (DRL) uses neural networks to approximate the policy and/or value functions. This enables the agent to handle large and complex state spaces, generalize across unseen situations, and scale to environments where tabular methods would be infeasible.

DQN is an off-policy algorithm that learns a Q-function to estimate the expected return of each action in a given state. While effective in many discrete action tasks, DQN can be unstable and prone to overestimation, especially when the state-action space is large or the environment has complex dynamics [8]. In *Nine Men's Morris*, the three-phase structure creates a shifting strategic landscape, making it challenging for a value-based agent to generalize across phases.

PPO, in contrast, is an on-policy, actor-critic algorithm that directly optimizes the policy. Its clipped objective prevents overly large updates, improving stability when strategies or optimal moves change across phases [7]. This makes PPO well-suited for our game, where the agent must adapt to different stages of play, although additional techniques such as action masking are required to handle the large action space efficiently.

To address the challenges posed by the game's mechanics, we augmented the standard PPO algorithm to incorporate **action masking**. In our integer-encoded action space of $25^3 = 15,625$ possible actions, the vast majority of moves at any given state are illegal. Without guidance, a standard agent spends a great amount of training time attempting invalid moves. By utilizing the *MaskablePPO* implementation from Stable-Baselines3 [9], we "mask out" illegal actions by setting their probabilities to zero before the softmax layer.

This combination of PPO's stable convergence and action masking provided the agent with a clear advantage: it removed the need to learn the basic rules of piece movement through trial and error, allowing the neural network to focus immediately on high-level strategy and board positioning throughout training.

B. Model Configuration

The policy is represented by a feedforward actor-critic architecture with separate network heads for the policy π and value function V. Each network consists of two fully connected hidden layers with 256 neurons per layer, using hyperbolic tangent (tanh) activation functions.

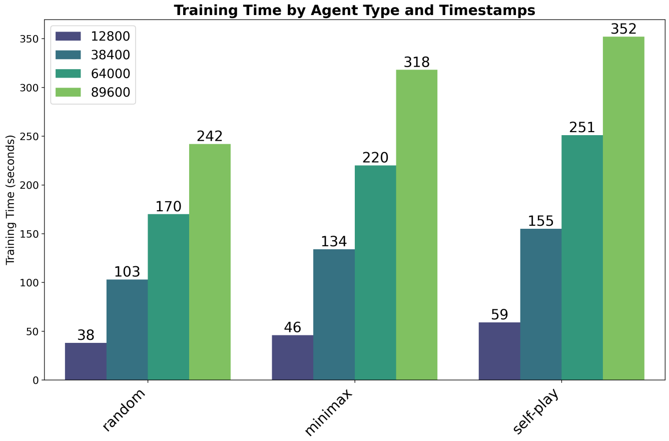
The training hyperparameters were selected based on typical PPO configurations for discrete action spaces, with adjustments to accommodate the game's strategic depth. We use a learning rate of 3×10^{-4}, a discount factor $\gamma = 0.99$ to emphasize long-term planning, and a Generalized Advantage Estimation parameter $\lambda = 0.95$. Each policy update collects $N_{\text{steps}} = 512$ environment steps, which are then processed in minibatches of size 64 over 3 optimization epochs. To maintain training stability, we apply gradient clipping with a maximum norm of 0.5 and enforce a PPO clipping range of 0.2. An entropy coefficient of 0.01 encourages exploration while preventing excessive randomness in the learned policy. All the models were trained in cloud(Kaggle) using GPU.

C. Choice of Training Opponents

As described before, one of the objectives of the paper is to understand how the choice of training opponent affects the agent. The model will be trained against the 3 opponents that can be characterized in the following way:

1) **Fixed Opponents:** The agent trains exclusively against a single, static opponent. We consider two variants of increasing complexity.
   - A **Random Agent**, which selects uniformly from all legal moves. It serves as a controlled baseline, used to examine if the agent can learn the absolute basics of the game when the opponent is unpredictable, makes no strategic sense and provides minimal resistance.
   - A **Weak Minimax Agent** (limited depth=1, 50% random moves). As it uses the Minimax logic together with heuristics, it is more principled than the pure Random Agent.
2) **Self-Play:** The agent trains exclusively against its recent (frozen) policy that is periodically updated. We make sure the policy is updated only periodically, to avoid the moving target problem of an opponent that changes its behavior each time step, as it can harm the convergence stability of our agent [10]. The periodical updates ensure a stable dynamic target: once the agent consistently beats its past self, the opponent is updated, forcing the agent to improve.

On the following visualization we can see time to train the agents against all 3 opponents for 25, 75, 125 and 175 episodes (1 episode is 512 steps) - what can be observed is that increasing the complexity of opponent, as expected also increases training time:



IV. Results

All three trained models were evaluated against four distinct baselines of increasing difficulty: (1) a completely random agent, and (2-4) minimax-based agents with limited search depth (1, 2, and 3 respectively), each making 50% random moves. The baselines are labeled as Random, Apprentice (depth=1), Adventurer (depth=2), and Knight (depth=3). Performance was measured across 1,000 games for each configuration, with results reported in the following table as (winrate percentage : average game length in moves).

Trained Against	Steps	Random	Apprentice	Adventurer	Knight
Random	12800	74.7 : 52.8	41.9 : 43.6	26.1 : 45.2	13.5 : 41.6
	38400	99.2 : 30.5	86.8 : 34.7	57.1 : 43.6	52.4 : 40.6
	64000	99.3 : 28.6	89.8 : 34.3	62.3 : 42.2	54.6 : 40.8
	89600	99.9 : 27.7	91.0 : 33.8	65.3 : 42.5	57.6 : 41.0
Minimax	12800	82.1 : 45.4	75.9 : 37.4	52.7 : 44.9	40.0 : 43.5
	38400	99.3 : 28.6	96.8 : 26.8	80.4 : 37.9	68.7 : 38.2
	64000	99.4 : 29.4	97.7 : 27.1	75.5 : 38.3	66.4 : 38.9
	89600	99.9 : 30.4	96.7 : 27.4	81.0 : 38.4	69.3 : 37.0
Selfplay	12800	81.1 : 48.0	33.4 : 44.0	16.4 : 44.8	11.3 : 42.0
	38400	98.7 : 30.4	73.8 : 33.8	60.8 : 41.8	51.0 : 41.3
	64000	98.8 : 29.0	84.4 : 32.7	71.8 : 38.5	57.6 : 38.0
	89600	99.8 : 29.1	79.9 : 32.3	75.4 : 37.0	63.8 : 38.4

Performance Comparison with Minimax Agents

The results demonstrate that RL agents can achieve performance levels comparable to classical search-based minimax agents. Across all training regimes, agents show consistent improvement with increased training duration, reaching win rates exceeding 99% against random opponents and 75%+ against Apprentice and Adventurer-level minimax by 89,600 steps. Most notably, agents trained against minimax and through self-play achieve win rates of 69.3% and 63.8% respectively against the Knight baseline (depth=3), demonstrating that learned policies can compete with moderately deep search-based approaches. The progressive improvement across increasingly difficult opponents confirms that the RL agents are learning genuine strategic competence rather than exploiting weaknesses in specific opponents.

Impact of Training Regime on Learning Efficiency. The choice of training opponent significantly affects both learning speed and final performance. Training against the minimax baseline produces the strongest overall results, with the agent achieving 81.0% win rate against Adventurer and 69.3% against Knight at 89,600 steps. This represents a substantial improvement over training against random opponents (65.3% and 57.6% respectively), which can be attributed to the structured nature of minimax play: even with 50% random moves, the deterministic strategic decisions provide consistent patterns from which the agent can extract useful policies.

Surprisingly, self-play does not dominate as initially hypothesized. While self-play eventually achieves competitive performance (75.4% against Adventurer, 63.8% against Knight), it lags behind minimax-trained agents. We attribute this to several possible factors: (1) the frozen version used was not stable/sensible enough yet at the time of training, (2) the opponent update frequency which we used (every 10% of training steps) may not be optimally calibrated for this domain; or (3) self-play may require significantly longer training to realize its theoretical advantages. Notably, self-play shows the strongest continued improvement trajectory in later training stages, suggesting it may eventually surpass fixed-opponent training given sufficient computational resources.

V. Possible Improvements and Future Work

While the current results provide insight into the impact of training opponent selection, several promising directions remain for developing more effective training configurations and stronger agents:

1) **Extended Training and Curriculum Learning:** The observed trajectory of self-play performance suggests continued improvement beyond our training horizon. However, computational constraints limited training duration due to evaluation depth - minimax agents with deeper search require substantial computation time per move. Future work should explore significantly longer training runs (500,000+ steps) to test the hypothesis that self-play eventually surpasses fixed-opponent methods. Additionally, curriculum learning approaches that gradually transition from weaker to stronger opponents could combine the early learning efficiency of structured baselines with the asymptotic advantages of self-play.

2) **Refined Reward Structure:** The current reward system (+1/-1 for piece capture/loss) effectively guides learning but treats all captures equivalently. This design does not distinguish between captures that occur mid-game and those that lead directly to terminal victory or defeat. Incorporating a substantial terminal reward (e.g., +10 for winning, -10 for losing) could provide clearer credit assignment, helping the agent better identify which sequences of piece exchanges lead to game-winning positions rather than merely local advantages.

3) **Action Space Optimization:** The integer-encoded action space of $25^3 = 15,625$ possible actions remains highly sparse even with action masking. Future research could investigate hierarchical action decomposition, or learned action embeddings. Reducing the output dimensionality could improve sample efficiency and enable the policy network to focus representational capacity on strategic patterns rather than navigating the large action space.

4) **Phase-Specific Policy Components:** Given the distinct strategic characteristics of the placement, movement, and flying phases, architectures with phase-conditioned policy heads or separate sub-policies could potentially improve performance. Such approaches would allow the agent to develop specialized strategies for each phase while still maintaining coherent overall gameplay.

VI. Conclusion

This study demonstrates that reinforcement learning agents can achieve performance comparable to classical search-based approaches in Nine Men's Morris. Our PPO-based agent with action masking reached competitive winrates against Minimax opponents, confirming that learned policies can compete effectively with traditional game-playing algorithms.

Regarding training regime selection, our findings reveal that opponent choice significantly impacts both learning efficiency and final performance. Training against structured Minimax opponents produced superior results compared to random-opponent training, yielding 81.0% win rate against Adventurer-level opponents versus 65.3% for random-trained agents. While self-play initially lagged behind fixed-opponent methods, its stronger improvement trajectory in later training stages suggests it may eventually surpass structured baselines given sufficient computational resources.

These results underscore the importance of opponent selection in reinforcement learning for strategic games, particularly in resource-constrained settings where moderately challenging structured opponents provide the most effective training signal.

References

[1] R. Gasser, "Solving nine men's morris," *Games of No Limit*, vol. 29, pp. 101–107, 1996.

[2] M. Ćetković, D. Tupkalenko, and D. Gashi, "Nine men's morris with minimax algorithm," https://github.com/marijacetkovic/mill_project, 2025, accessed: 2026-01-09.

[3] D. Šoberl, "Famnit gym environment," https://github.com/DomenSoberlFamnit/famnit-gym, 2025, accessed: 2026-01-09.

[4] T. Farrell *et al.*, "Gymnasium: A standard api for reinforcement learning environments," https://gymnasium.farama.org/, 2023, accessed: 2026-01-09.

[5] J. Terry, B. Black, N. Grammel, M. Jayakumar, A. Hari, R. Sullivan, L. S. Santos, C. Dieffendahl, C. Horsch, R. Perez-Vicente *et al.*, "Pettingzoo: Gym for multi-agent reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 15 032–15 043, 2021.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[8] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, 2016, arXiv:1509.06461. [Online]. Available: https://arxiv.org/pdf/1509.06461.pdf

[9] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable baselines3," https://github.com/DLR-RM/stable-baselines3, 2021, accessed: 2026-01-09.

[10] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: a survey," *Artificial Intelligence Review*, vol. 55, pp. 895–943, 2022.