Robo Arm

Frederik Schwarz
Informatik
Hof-University

***Abstract*—This work presents a robotic system capable of autonomously playing Nine Men's Morris by integrating computer vision and strategic decision-making. The system employs a robotic manipulator to physically move pieces on the board while leveraging YOLO-based visual perception to detect the current board state in real time. A custom-trained model evaluates the game state and selects optimal moves, enabling the robot to play strategically against human or simulated opponents. We detail the design of the perception pipeline, the interfacing of the robot arm with the board, and the training of the game model, emphasizing how these components interact to achieve seamless gameplay.**

A. State Encoding

To prepare the game data, the positions of the game pieces are represented using one-hot encoding. In addition to the position-related information, global features are integrated into the state vector. These include the current game phase (“placing”, “moving”, “jumping”), which is also one-hot encoded, the number of pieces still to be placed in normalized form, and binary indicators specifying whether a piece must be removed in the current turn and whether the game is currently in the jumping phase.

The encoding of the game phase is deliberately redundant in order to explicitly convey to the model that the movement rules change once a player has only three pieces remaining on the board. In this way, the game state is always coupled to two inputs: first, information about whether pieces still need to be placed (placing phase), and second, the explicit state encoding of the current game phase. This results in two separate input representations: a tensor-shaped representation of dimension (3, 24) for modeling the piece positions, and a vector of length 11 describing the global state features.

The output space of the model is defined as a one-dimensional array of length 24×25. Each possible combination of source and target positions is represented by a distinct probability. Indices 0 to 23 correspond to the actual positions on the game board, while index 24 is used as a placeholder when no source or target position exists, such as during the placing or removal of a piece. The corresponding index in the output array is computed as $\text{index} = \text{source} \times 24 + \text{target}$ where source denotes the starting position and target the destination position.

The eleventh global feature, indicating whether a stone must be removed, was introduced at a later stage of development. As a consequence, the special index 24 in the output space became overloaded to also represent removal actions, which is a suboptimal but empirically functional design choice.

B. Neural Network Architecture

The encoded input information is fed into separate input layers. The positional information of the game pieces is flattened beforehand, as the chosen architecture is a simple feed-forward network that does not provide inherent support for structured or sequential data.

After processing in the respective input layers, the resulting outputs are concatenated and subsequently passed through two hidden layers. Each hidden layer consists of 256 neurons. Since no well-founded prior knowledge regarding the optimal dimensionality of the feature representations was available, the number of neurons was chosen heuristically.

In the final processing step, a policy head is defined that outputs logits. These logits can then be interpreted by the action mapper to derive concrete actions. In addition, a separate value head was implemented for training purposes. This head serves to approximate the state value or, respectively, the advantage of the current game state. Accordingly, the output value should be close to 1 when the model is in a near-winning situation and close to −1 when a loss is imminent.

The value head first reduces the representation via an additional hidden layer with 128 neurons and then projects it onto a scalar output. The hyperbolic tangent function (tanh) is used as the activation function to explicitly constrain the output range to the interval $[-1, 1]$. For all remaining layers, the ReLU activation function is employed.

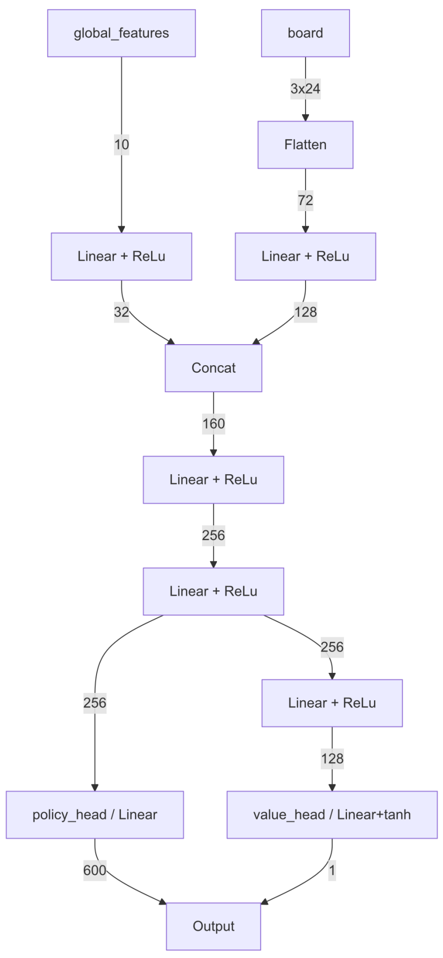

Fig. 1. Model Architektur

a) *Possible Model Architecture Improvements:* Currently, the model is implemented as a simple feedforward network. However, since the game of Nine Men’s Morris is essentially a graph and capturing graph-based relationships may contribute to a better modeling of the game structure, the use of a Graph Neural Network (GNN) could be advantageous.

b) *Feed Forward Neural Network:* A feedforward neural network is an artificial neural network in which information processing is strictly unidirectional. The input values are multiplied by weighting factors layer by layer and transformed into output values.

The feedforward structure is a fundamental prerequisite for the application of the backpropagation algorithm and was one of the first network architectures to be implemented. [1]

A deliberately simple neural network architecture was chosen to demonstrate the fundamental capabilities of neural networks and to assess the minimum architectural complexity required to achieve reasonable performance, without relying on more elaborate or externally proposed model designs.

c) *Multi-layer perceptron:* A multilayer perceptron (MLP) refers to a class of modern feedforward neural networks (as explained above) consisting of fully connected neurons organized into multiple layers and employing nonlinear activation functions. A defining characteristic of MLPs is their ability to model and classify data that are not linearly separable.

Multilayer perceptrons represent a fundamental architecture in deep learning and are applied across a wide range of different domains. [2]

d) *Graph Neural Networks(GNN):* Graph Neural Networks (GNNs) are a specialized class of artificial neural networks designed for tasks in which the input data are represented in the form of graphs. [3]

C. Training

The training loop comprised several stages. It began with the agent playing a game and generating rewards at each step. Illegal moves were masked and penalized, while positive rewards were assigned for actions such as blocking mills, winning, gaining a positional advantage, forming mills (removing opponent pieces), and creating potential mills. Experience was accumulated until a predefined threshold was reached, at which point backpropagation was performed and the updated weights were applied. The techniques described below were employed to implement these training and reward mechanisms a) *Reinforcement Learning:* Reinforcement Learning is a subfield of machine learning and optimal control that addresses the question of how an intelligent agent selects actions in a dynamic environment in order to maximize a long-term reward signal. In contrast to supervised and unsupervised learning, which are based on the analysis of labeled and unlabeled data respectively, learning in reinforcement learning occurs through direct interaction between the agent and its environment. The agent receives feedback in the form of rewards or penalties, on the basis of which it gradually learns an optimal action strategy (policy). [4]

b) *Self-play:* Self-play is a paradigm in which an agent plays against itself in order to improve its strategy. This allows us to train our Nine Men's Morris AI without using a dataset or requiring any manual intervention during training. [5]

c) *Actor-Critic:* The actor–critic algorithm represents a class of reinforcement learning methods that combine two central components: the actor, which selects actions, and the critic, which evaluates these decisions. This dual structure makes the agent's learning process more efficient, as decision-making and feedback are balanced. While the actor learns how to make decisions, the critic assesses their quality. This enables the agent to explore new actions while simultaneously building on previously acquired experience, resulting in a more stable and effective learning process. The policy corresponds to the actor, which selects actions, whereas the value function represents the critic, which evaluates the quality of these actions. [6]

d) *GAE:* Generalized Advantage Estimation (GAE) is used to compute the advantage function, providing a balance between bias and variance in policy gradient updates. By weighting temporal-difference errors over multiple time steps, GAE produces more stable and efficient learning signals. This allows the agent to learn more effectively from sparse or delayed rewards while reducing variance in the gradient estimates. [7]

e) *Entropy Regularization:* Entropy regularization is a method for regularizing model outputs, in which an additional term is added to the loss function that encourages the model to produce outputs with higher entropy. Entropy, in the sense of information theory, quantifies the uncertainty or randomness of a probability distribution. By maximizing the entropy of the output distribution, the model is encouraged to make more diverse and less overconfident predictions. [8]

f) *Reward Shaping:* Reward shaping is a method in which additional rewards are introduced to accelerate an agent's learning. The goal is to guide the agent step by step in the right direction, especially in situations where the original reward occurs very rarely or sparsely. [9] In our case, the agent is guided such that gaining a game advantage and closing mills are encouraged through positive rewards, while illegal moves receive negative feedback.

It should be noted that this constitutes one of the most important strategies for enhancing agent performance. In this setup, the agent is incentivized to maximize the rewards it receives rather than necessarily achieving victory in the game.

g) *Gradient Clipping:* Gradient clipping refers to a technique aimed at ensuring the stability of training neural networks by limiting the magnitude of the gradients. During the backpropagation process, gradients are computed, which determine the direction and strength of the weight updates in the network. However, if excessively large gradients occur —referred to as "gradient exploding"—the resulting weight changes can cause numerical instabilities, such as the occurrence of NaN values or overflow errors. Gradient clipping addresses this problem by constraining the norm of the gradients to a predefined maximum value. In this way, it ensures that the updates of the model parameters remain controlled without significantly limiting the network's learning capability, thereby contributing substantially to a stable and efficient training process. [10]

h) *Temperature Scaling:* Temperature scaling is used to calibrate the probabilities output by the policy network. A learnable temperature parameter T is applied to the softmax outputs, which dampens overly confident probabilities and represents uncertainties more realistically. In the game of

Nine Men's Morris, this prevents the agent from favoring incorrect moves with excessive confidence, encourages the exploration of alternative strategies, and stabilizes training without altering the actual action decisions. [11]

i) *ε-Greedy Exploration:* The ε-greedy algorithm is a strategy that allows the agent to select actions in order to balance exploration (trying out new or infrequently chosen actions) and exploitation (leveraging already known, promising actions). With a probability of ε, the agent selects a random action (exploration), while with a probability of 1−ε, it chooses the action with the highest estimated action value (exploitation). In this way, the agent can both gain new insights and efficiently utilize known reward potentials. [12]

References

[1] "Feedforward neural network." Accessed: Jan. 14, 2026. [Online]. Available: https://en.wikipedia.org/wiki/Feedforward_neural_network

[2] "Multilayer perceptron." Accessed: Jan. 14, 2026. [Online]. Available: https://en.wikipedia.org/wiki/Multilayer_perceptron

[3] "Graph neural network." Accessed: Jan. 14, 2026. [Online]. Available: https://en.wikipedia.org/wiki/Graph_neural_network

[4] "Reinforcement learning." Accessed: Jan. 14, 2026. [Online]. Available: https://en.wikipedia.org/wiki/Reinforcement_learning

[5] "Self-play." Accessed: Jan. 14, 2026. [Online]. Available: https://en.wikipedia.org/wiki/Self-play

[6] "Actor-Critic Algorithm in Reinforcement Learning." Accessed: Jan. 14, 2026. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/actor-critic-algorithm-in-reinforcement-learning/

[7] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-Dimensional Continuous Control Using Generalized Advantage Estimation." [Online]. Available: https://arxiv.org/abs/1506.02438

[8] "Definition and Significance of Entropy Regularization." Accessed: Jan. 14, 2026. [Online]. Available: https://www.numberanalytics.com/blog/entropy-regularization-machine-learning

[9] "Epsilon-Greedy Algorithm in Reinforcement Learning." Accessed: Jan. 14, 2026. [Online]. Available: https://rljclub.github.io/posts/reward-shaping/

[10] "Understanding Gradient Clipping." Accessed: Jan. 14, 2026. [Online]. Available: https://www.geeksforgeeks.org/deep-learning/understanding-gradient-clipping/

[11] "What Is Temperature Scaling for LLM Calibration?." Accessed: Jan. 14, 2026. [Online]. Available: https://markaicode.com/temperature-scaling-calibrate-llm-confidence-scores/

[12] "Epsilon-Greedy Algorithm in Reinforcement Learning." Accessed: Jan. 14, 2026. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/epsilon-greedy-algorithm-in-reinforcement-learning/