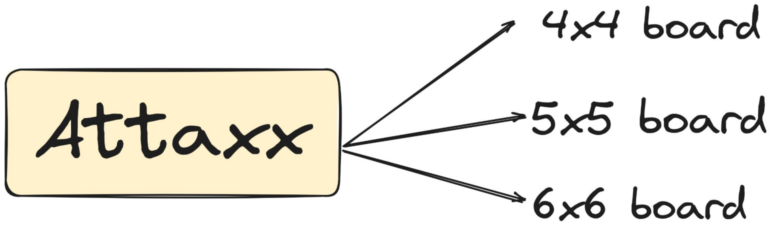
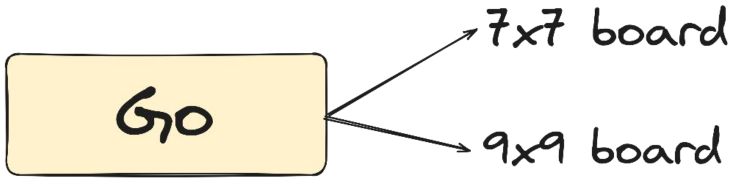
Project 2 – Develop an Alpha Zero Game Player

Work done by:
Bárbara Santos 202108573
David Scarin 202108314
Inês Cardoso 202107268
Pedro Sousa 202108383

The project

- In this project we were asked to develop an AlphaZero Algorithm-powered game player and evaluate its performance across five distinct scenarios:

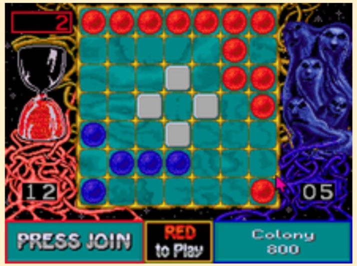


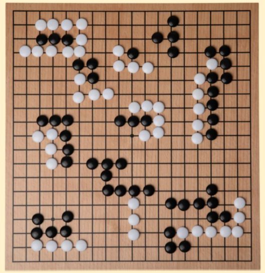


Attaxx & Go

Attaxx is a strategy game, it features a two-player competition on a seven-by-seven square grid, with the goal of one player achieving a majority of the pieces on the board by converting as many of their opponent's pieces as possible.

Go is a strategic game where the goal is to encircle a greater overall area of the board with one's stones compared to the opponent. Throughout the game players strategically place stones on the board to outline formations and potential territories.





Alpha Zero

- Alpha Zero is an algorithm introduced by DeepMind, starting from random play and given no domain knowledge except the game rules, a trained agent is capable of achieving superhuman level of performance, it completes this goal by combining a Monte Carlo Tree Search (MCTS) and a Neural Network in a policy iteration framework to achieve stable learning. Combining these elements an agent can then learn through self-play.

Monte Carlo Tree Search explores the search tree, where each node is a possible configuration of the board originated by a specific move. The data obtained is used in the training of the network in order to determine which moves lead to more favorable states.

The neural network attempts to learn, over time, what states eventually lead to wins (or losses).
In addition, learning the policy would give a good estimate of what the best action is from a given state.

Self play is when the agent plays games against itself and updates its policy based on the results. At the end of the iteration, the neural network is trained with the obtained training examples. The old and the new networks are pit against each other and the network is updated if it 'wins' according to a set threshold.

Alpha Zero-implementation

- Our implementation of Alpha Zero began with configuring the Monte Carlo Tree Search (MCTS) using the following parameters:

Prior Probability:

Each node stores the prior probability of the action that led to its creation. This probability is used during node expansion.

Dirichlet Noise:

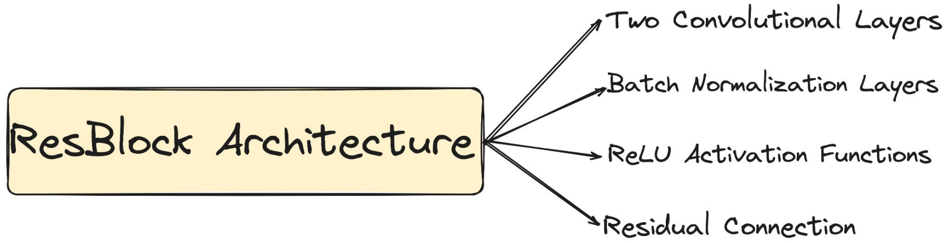
During node expansion, Dirichlet noise is applied to the predicted policy probabilities. This introduces a level of exploration, ensuring diversity in the selection of child nodes.

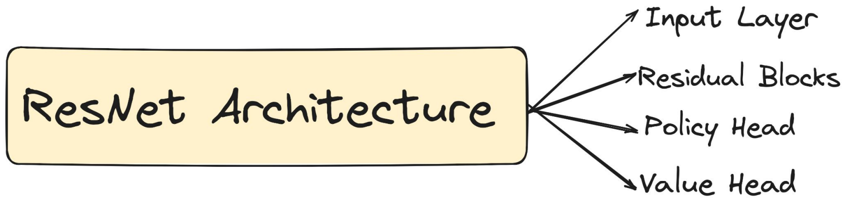
Neural Network Integration:

The MCTS class incorporates a neural network model to predict action probabilities and state values. These predictions guide the exploration-exploitation balance.

Alpha Zero-implementation

- Regarding the neural network, we decided to use a neural network architecture known as ResNet for the AlphaZero algorithm. This model is designed to take a game state as input and produce both a policy and a value.





Forward Method: Executes the forward pass through the residual block. Captures and enhances features in the input. Returns the output tensor after passing through the block.

Alpha Zero- overview

MCTS:

The MCTS class orchestrates the MCTS algorithm utilizing a neural network model to guide the exploration of the game tree. The search method performs batched MCTS, iteratively selecting, expanding, simulating and backpropagating to find the best action probabilities.

Neural Network Model (ResNet):

The ResNet class represents the neural network model used in AlphaZero. It takes a game state as input and outputs both a policy (probability distribution over actions) and a value (indicating the likelihood of winning). The model architecture includes residual blocks for effective feature learning.

Node Class

The Node class represents nodes in the Monte Carlo Treestoring information such as the state, action, player, and visit count. It includes methods for selection, expansion, and backpropagation within the MCTS process. The is_expanded method checks whether a node has been expanded.

Game-Specific Adjustments:

Game-specific conditions are considered, such as handling the "pass" action in Go.The game class is assumed to provide functionalities like getting the next state, changing perspectives, and determining valid moves.

Integration of Neural Network with MCTS:

The neural network model is integrated into the MCTS process to guide the search and enhance decision-making.

Alpha Zero-features

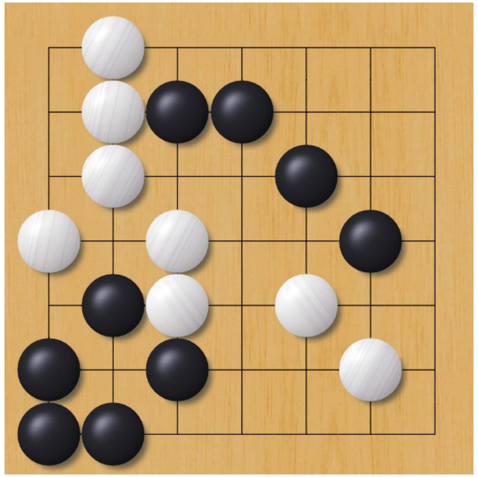
Data Augmentation

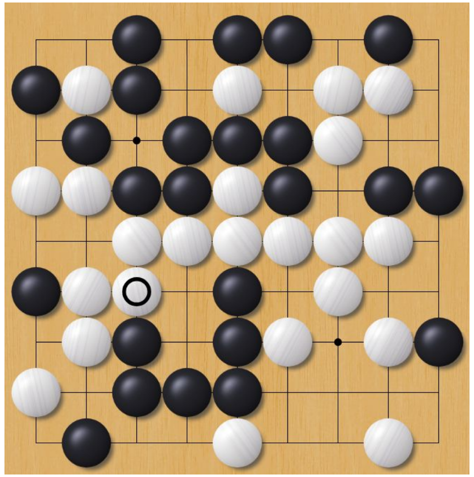
Simulated Annealing

Experience Replay

MCTS Node Caching

Results-Go

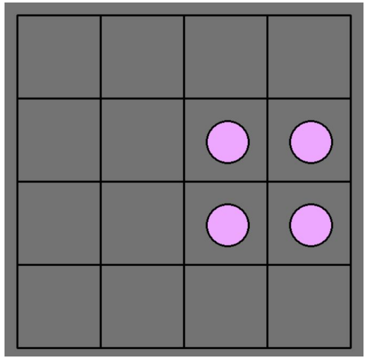

7x7

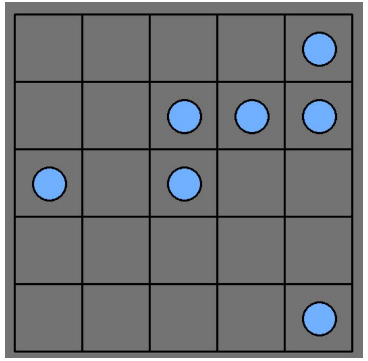

9x9

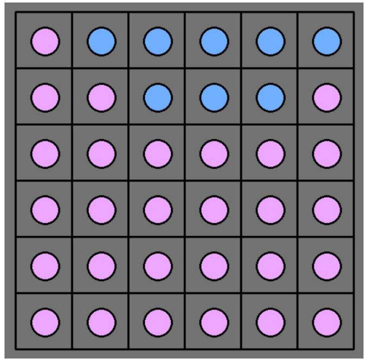
- ✓ Structured play
- ✓ Understanding of space
- ✓ Slight understanding of life and death

- ✗ Sometimes doesn’t play critical moves
- ✗ Doesn’t fully understand if a group is already alive
- ✗ Plays inside granted territory

Results-Ataxx


4x4


5x5


6x6

- ✓ Effective expansion strategy
- ✓ Good control of the board
- ✓ Adaptable to opponent's moves

- ✘ Occasionally misses optimal play
- ✘ Does not always predict opponent's advances

Conclusion & Next Steps

Conclusion:

- Successful integration of the games and Alpha Zero
- Development of new features

Next Steps:

- Optimization of both games and algorithms
- Implementation of more games
- Deepening and fine tuning the training of our models