Reinforcement Learning with AlphaZero

In the area of artificial intelligence, the creation of AlphaGo, developed by DeepMind in 2014, was marked as a groundbreaking project. Knowing nothing but the rules of the game of Go, by 2016 it managed to beat in a 5-game duel one of the best Go players at the time, Lee Sedol; its successor, AlphaZero, which was created to be able to learn and master any board game, set a new standard on the status quo of AI. In this project, we aimed to develop an implementation of AlphaZero from scratch, further modifying it, and train multiple models across two different board games, Go and Attaxx.

To the standard development of a generalized AlphaZero algorithm, we implemented:

- **Data Augmentation**: As both games are invariant to rotation and translation, we added a toggleable feature that would add to training memory all the possible transformations of every state, in order to produce more data with less self-playing, and to further enhance its gameplay knowledge.

- **Experience Replay**: By storing and reusing past self-play experiences, we will make sure the model revisits past game scenarios, adapting to new strategies as it learns, and minimizing the risk of overfitting.

- **Simulated Annealing**: Inspired by the process of annealing in metallurgy, was integrated into our AlphaZero implementation to optimize the decision-making process. By gradually reducing the exploration rate over time, the algorithm is encouraged to explore a wide range of strategies in the early stages of learning and then focus on exploiting the best ones that were learned. This approach helps in avoiding local optima and encourages the models to understand in depth the game mechanics as it tries new strategies out.

- **MCTS Node Caching**: In addition to the methods already implemented, we developed a caching mechanism for the results of the Monte Carlo Tree Search within our AlphaZero framework. This technique involves saving the outcomes of previous MCTS simulations and reusing them in future decision-making processes. By doing so, the algorithm can significantly accelerate its search speed, as it avoids redundant tree searches for states that have been previously encountered.

In conclusion, we successfully developed our own AlphaZero training system from scratch, adapted to both Go and Attaxx, as well as implemented various extra strategies regarding efficient data usage. The biggest challenge encountered during the development of this project was the large dependence on time and computing resources. Due to these constraints, we aimed to implement the strategies described earlier in order to maximize the efficiency of our agent's training.

The development of this project was satisfying as we could see clear progress along its creation, and the results exceeded our expectations. We obtained robust and efficient models that could efficiently play the games we were proposed to develop, and this success reaffirms our dedication to the development of AI.

meow >:3