AlphaZero Implementation: Main Features

Project of Laboratory of Artificial Intelligence and Data Science

André Sousa • Antónia Brito • António Cardoso • Paulo Silva

Self-Play Games in Parallel Pool

To optimize the training efficiency of the AlphaZero algorithm execution, a parallel game pool was introduced, alongside a customized size parameter denoted as *G*. This design permits the simultaneous execution of up to *G* games during training. In contrast to alternative approaches for parallelizing the concurrent execution of multiple self-play games, such as the one demonstrated in FreeCodeCamp's AlphaZero implementation video (link), our implementation demonstrates superior performance. This advantage arises from the avoidance of waiting for the completion of all *G* games before initiating a new set. Instead, upon the conclusion of a game within the total running set, its results are stored, facilitating the initializing of a new game in its place.

Temperature Parameter for Exploratory Search

Drawing inspiration from simulated annealing principles, a temperature parameter was incorporated into the algorithm. This parameter follows a sigmoidal decay, as a function of the current AlphaZero iteration count. This dynamic temperature regulation enables the model to employ an exploratory strategy in the initial iterations, progressively transitioning towards learning optimal moves and playing strategies for the specific game. The integration of this temperature parameter enriches the learning process, striking an additional balance between exploration and exploitation on top of the MCTS' one.

One Model, Multiple Board Sizes

Each model developed within this project is specifically tailored to the individual game and board size in play. Consequently, a single model cannot be utilized across varying board sizes, limiting adaptability. To go around this problem, a feature was devised to address this limitation, wherein the model is trained on data from self-play games of various board sizes. To facilitate this, since the network input remains fixed, it was necessary to add padding to the board representations with -1 values. This adaptation allows the model to learn how to adjust probability values for different board sizes.

Data Augmentation

Recognizing the rotational and symmetrical invariance of the game boards considered for this project (Ataxx and Go), combinations of these transformations were applied to generate diverse board configurations, complete with corresponding action probabilities and board evaluations. This approach augments the dataset, reducing the requirement for extensive self-play games while still accumulating a substantial volume of valuable data points.

MCTS Tree Reutilization

In pursuit of achieving more profound Monte Carlo Tree Searches (MCTS) and conserving computational resources, a mechanism for reutilizing previously computed nodes from the latest MCTS was implemented. This approach optimizes the calculation of subsequent moves by leveraging existing nodes and values, which results in skipping redundant tree expansions and reducing computational costs.

Mitigating Overfitting and Increasing MCTS Exploration with Noise

To mitigate overfitting and amplify exploration within AlphaZero's self-play games, a controlled noise injection into the probability distribution of each Monte Carlo Tree Search root was employed. Originating from a Gaussian distribution with a mean of 0 and a standard deviation of 0.01, this intentional introduction of noise at the MCTS tree root leads to less deterministic outcomes, enriching the spectrum of encountered game situations for enhanced learning. This strategic noise application prevents excessive fixation on specific patterns, promoting a more adaptable and less overfitted model.