AlphaZero for Connect 4

Ishtdeep Singh, Wei Ren Gan, Aayush Chudgar
is30@rice.edu, wg21@rice.edu, anc3@rice.edu
Rice University

Abstract

In this project, we want to explore the AlphaZero [7] Reinforcement Learning (RL) algorithm which was developed by Deepmind in 2017 by applying it to the game of Connect 4. Connect 4 is a compelling two-player slot and disc game where players have the opportunity to develop and implement strategies. We studied AlphaZero, a mixture of planning and RL, to find how learning occurs and the main strengths of AlphaZero as an algorithm.

1 Introduction

Connect 4 is a game that was invented by Milton Bradley Company in 1974. The objective of the game is to get four of the same colored discs in a row, either vertical, horizontal or diagonal. The rules of the game is as follows [1]:

- Players must alternate turns, and only one disc can be dropped in each turn.
- On your turn, drop one of your colored discs from the top into any of the seven slots.
- The game ends when there is a 4-in-a-row or a stalemate.

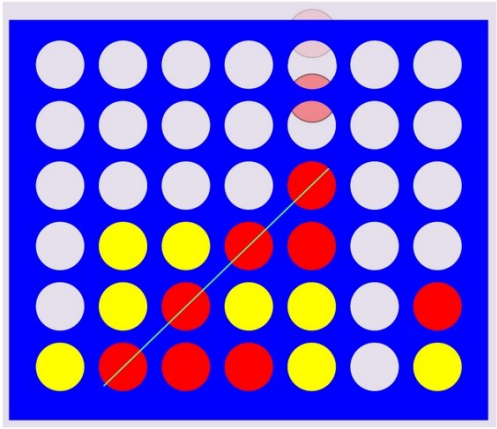

Figure 1: Example Connect 4 Board - Winning Game [2]

Connect 4 might seem to be a simple game, but its state complexity suggests that there are about 10^{12} positions possible in a 6 x 7 board. On average from each position, there are more than 6 moves possible.

Due to the large variety of possible final boards, players are able to take numerous different actions to win. The motivation behind our project is to train virtual agents to find the optimal policy in

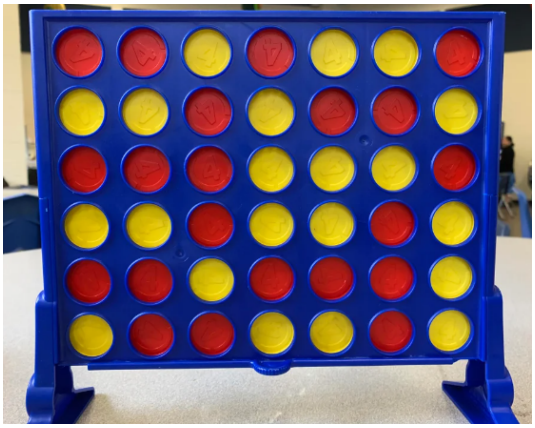

Figure 2: Example Connect 4 Board - Tie Game [3]

large state-space. We will use the AlphaZero [7] algorithm and perform evaluation on Connect 4. AlphaZero is a mixture of planning and RL. It uses a neural network which provides the value function which gives an estimation of the probability of winning from the given position, and the policy which is a probability distribution across all the legal actions. It uses Monte Carlo Tree Search (MCTS) and rolls out multiple times from the current position to figure out which move is the best and then uses the Upper Confidence Bound (UCB) to pick the actions. AlphaZero has found a lot of success in complex and lengthy board games (Board games in which the reward is often delayed until the end) like Chess, Go and Shogi.

2 Related works

A Stanford paper presents a reinforcement learning approach to the classic two-player slot and disc game Connect Four. They survey two reinforcement learning techniques—Sarsa and Q-learning— to train an agent to play Connect Four using an optimal strategy for the game. They studied how varying exploration rate and rewards models affects performance by both algorithms. Ultimately, they did not find a significant difference between the two algorithms, as both recorded strikingly similar win percentages against themselves and against the opposing algorithm.[5]

Cologne University applies temporal difference learning (TD), a well-known variant of the reinforcement learning approach, in combination with n-tuple networks to the game Connect-4. Their agent is trained just by self-play. It is able, for the first time, to consistently beat the optimal-playing Minimax agent(in game situations where a win is possible). The n-tuple network induces a mighty feature space: It is not necessary to design certain features, but the agent learns to select the right ones. They believe that the n-tuple network is an important ingredient for overall success and identify several aspects that are relevant for achieving high-quality results. The architecture is sufficiently general to be applied to similar reinforcement learning tasks as well.[10]

In late 2017 DeepMind introduced AlphaZero, which uses no human knowledge and purely learns through self play. They used the MCTS algorithm to set the target policy and trained the value function and policy towards this target. They trained AlphaZero from scratch and trained it till superhuman performance and defeated the best chess, GO and Shogi playing algorithms at the time of release. They improved the neural network for the value function of AlphaZero, till the value function itself had superhuman performance and was rated over 3000 elo (best human experts have an elo less than 2900).

3 Model setup

Our project attempts to find an optimal Connect Four playing strategy given a Connect Four board. We define an optimal Connect Four playing strategy as the sequence of turn actions that maximizes the probability of a player winning a single Connect Four game.

We implemented the AlphaZero Algorithm for this task as it an algorithm which can take advantage of the partially observed model of the environment.

3.1 Action Space

The action space for Connect Four is relatively small. Each player has at most seven possible actions that they can take at any given state. Therefore an action is defined as dropping a piece into one of the seven columns on the board. Because the number of actions a player may take at a given state depends on how many columns on the board are full, we calculate the number of actions possible at a given state as:

Actions Possible = 7 - C

C = Number of full columns at current state

3.2 State Space

The state space for Connect Four is considerably larger than the action space. A state in Connect Four is defined as the board with played pieces that a player sees. A very rough upper bound for the number of possible states of a connect four game can be calculated by taking into account that each position on the board can either be free, a player one piece or a player two piece. Then because the board size is 6 x 7, we get an upper bound of 3^{42}. However, this calculation does not take into consideration that possible boards are illegal due to gravity and the rules of the game. The best lower bound on the number of possible positions has been calculated by a computer program to be around 10^{12}.

4 Algorithm

In AlphaZero there is a neural network which predicts the value function v and the policy pi given a representation of the board. The value function indicates the probability of the agent winning the current game. The policy is a probability distribution over all the legal actions. For training this we create a replay data structure which stores the most recent 200000 states, actions, rewards and PI which comes from Monte Carlo Tree Search (MCTS). This buffer is filled with the examples that the agent has seen. The state (s) is represented by a (3,6,7) tensor. Where Each matrix represents a portion of the board like player 1's pieces, player 2's pieces, empty spots.

For a board state s, AlphaZero first executes a set amount of MCTS [8] rollouts (in our case we trained it with 100 rollouts) per turn. A rollout is defined as playing a complete game from the current state using the agent's current policy (defined by the neural network). In these rollouts, we maintain a tree sort of structure which has a root s, and its children represent other states which are reached by playing moves given by the policy and UCB. The move selection for these rollouts is done using Upper Confidence Bound which ensures that we balance both exploration and exploitation. [8] During the rollouts, we keep track of the plays which led the agent to win and which led to a loss. After performing the set number of rollouts, we form a probability distribution over all the actions using the information of the rollouts. This probability distribution is PI. We then pick the action according to this distribution and store (state (s) , action (a) , pi, result (r)) in the replays. We use this replay buffer like a dataset and optimize for the following objective :

$$(f(s) - r)^2 - \sum_{i}^{C} t_i log(f_p(s))$$

where f(s) is the value function prediction of the network and r is the actual outcome that took place at the end of the corresponding episode. The second term tries to reduce the KL divergence between the policy predicted by the neural network $f_p(s)$ which is pi and the policy that is devised by the MCTS PI. For training this, we train the network once every set number of iterations for a set number of epochs (100 in our case).

The intuition behind these updates is quite straight forward. Monte Carlo tree search is proven to give the real state value function as the number of simulations tend to infinity. Since we have limited

resources we search fewer times but not randomly and with the guidance of the approximate value function by our neural network. Looking ahead and playing multiple games from the current state will give a more accurate prediction of the quality of the moves because the moves that are chosen in the MCTS search which led to victory multiple times would most likely be good moves (as the agent continues to learn). By looking ahead moves and then using our value function and policy function to make a prediction will lead to more accurate predictions because for example, if there is one move which would lead to victory then that move is more easier and obvious to spot near the end of the game rather than the beginning as the options towards the end are far fewer than the ones at the beginning.

4.1 Implementation details

For implementing this project we first had to implement the connect 4 environment. We built over the Kaggle connectX environment and made a simpler connect 4 environment. [4] We then implemented the AlphaZero algorithm referencing the deepmind paper and also building upon the generalized structure of AlphaZero provided by [9].

The neural network that we use has 4 CNN layers and 2 dense layers one for the policy and the other for the value function. Each CNN layer has 128 channels and 3 kernels. We use batchnormalization and the vanilla Adam optimizer with a learning rate of 0.001.

For further details please refer to our code where we explain the hyperparameters and the architecture.

5 Results

We trained our algorithm for 1000 episodes, updating the neural network after every 20 epochs. It took more than 20 hours on a 16 core, RTX 3060 system. To test the strength of our implementation of AlphaZero, we tested it against a random agent which chooses any legal action with equal probability. The results can be seen in Figure 3.

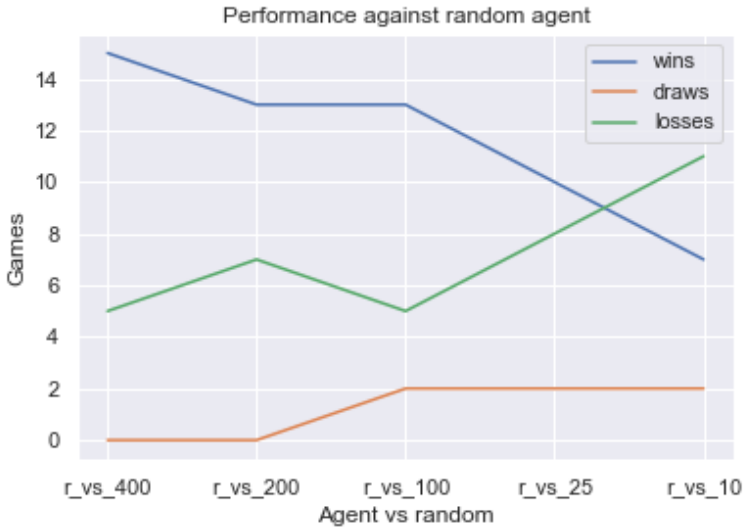

Figure 3: Our agent vs a random agent. The x axis tells us the number of MCTS simulations per move

This suggests that our implementation of AlphaZero version could defeat a random bot with ease as we increased the number of MCTS iterations. Thus this also suggests that the strength of our implemented algorithm is coming from the MCTS search part.

Further we tested how it fairs against versions of itself, which use the same neural network, but different number of MCTS simulations per move.

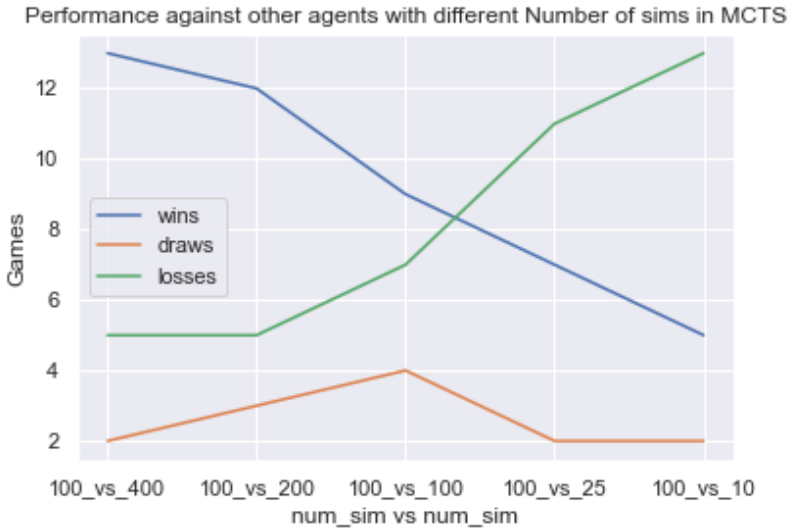

Figure 4: Our agent vs itself with different number of MCTS simulations per move

We see that as we increase the MCTS simulations, the strength increases and so does the win rate of the agent with higher number of simulations.

We also tested the strength of the neural network with puzzles. We made boards where the agent could win if it chose the right move. For these puzzles, we did not use the MCTS to get the best move. Instead, we used the neural network to test the understanding of the network. The value function usually gave the right answer while the policy network was usually wrong when choosing the move. This shows that we need to train a lot more to improve this algorithm.

6 Conclusion

The AlphaZero algorithm does work well with so few episodes but it relies on the MCTS simulations until the neural network improves. It is quite computationally expensive, as it takes about 20 hours to train for just a 1000 iterations. With just 1000 iterations it is able to defeat a random bot, and by increasing the number of MCTS simulations the performance is substantially improved.

7 Future Work

Implementing the AlphaZero algorithm was quite complex. In the future we would improve upon the gym environment that we created for Connect 4 and make it faster by using parallel processing. Due to the lack of computational resources we could not experiment a lot with the hyperparameters like the size of the neural network, learning rate, the exploration rate, etc. We would also train this algorithm for longer for 100,000 episodes so as to produce stronger agents. As part of our future scope, we plan to fully integrate Proximal Policy Optimization (PPO) [6] and other algorithms into our analysis and performances across various games such as Chain Reaction.

8 Appendix

8.1 Reproducing the result

- First set the hyperparameters in main and NNet model python files
- Run the main python file to train the algorithm
- For testing run the notebook ConnectX results

8.2 Contribution

8.2.1 Ishtdeep Singh

- Implemented the gym environment for Connect 4
- Implemented the MCTS algorithm of AlphaZero
- Implemented the neural network for the policy and value functions
- Compiled testing and results

8.2.2 Wei Ren Gan

-
- Written Abstract, Introduction, and other portions of the document

8.2.3 Aayush Chudgar

9 References

References

[1] URL: https://www.gamesver.com/the-rules-of-connect-4-according-to-m-bradley-hasbro/.

[2] URL: https://www.researchgate.net/figure/Traditional-Connect-Four-Here-Red-wins-with-four-coins-aligned-diagonally_fig1_258381798.

[3] URL: https://www.reddit.com/r/mildlyinfuriating/comments/aoh4fc/when_you_tie_at_connect_four/.

[4] Bovard Doerschuk-Tiberi Addison Howard. *Connect X*. 2020. URL: https://kaggle.com/competitions/connectx.

[5] Wopat & Koffman Alderton. *Reinforcement Learning for Connect Four*. 2019.

[6] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. DOI: 10.48550/ARXIV.1707.06347. URL: https://arxiv.org/abs/1707.06347.

[7] David Silver et al. *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm*. 2017. DOI: 10.48550/ARXIV.1712.01815. URL: https://arxiv.org/abs/1712.01815.

[8] R.S. Sutton and A.G. Barto. *Reinforcement Learning, second edition: An Introduction*. Adaptive Computation and Machine Learning series. MIT Press, 2018. ISBN: 9780262039246. URL: https://books.google.co.in/books?id=5s-MEAAAQBAJ.

[9] Shantanu Thakoor, Surag Nair, and Megha Jhunjhunwala. *Learning to play othello without human knowledge*. 2016.

[10] Markus Thill, Patrick Koch, and Wolfgang Konen. "Reinforcement Learning with N-tuples on the Game Connect-4". In: Sept. 2012. ISBN: 978-3-642-32936-4. DOI: 10.1007/978-3-642-32937-1_19.