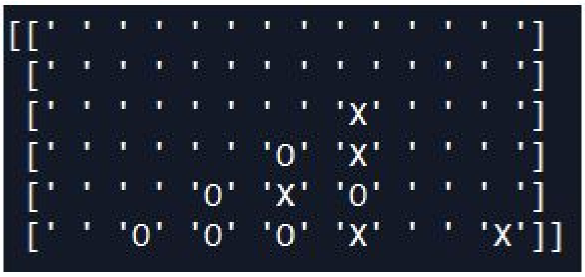
Akshitha Pothamshetty
116399326

Connect4 Game using AlphaZero algorithm

In this homework, I have trained an agent using the AlphaZero algorithm to play a Connect4 game.

In the architecture, A deep convolutional residual neural network, using PyTorch is used with architecture similar to AlphaZero to map an input Connect4 board state to its associated policy and value.The raw Connect4 board is encoded into a 6 by 7 by 3 matrix of 1’s and 0’s before input into the neural net, where the 3 channels each of board dimensions 6 by 7 encode the presence of “X”, “O” (1 being present and 0 being empty), and player to move (0 being “O” and 1 being “X”), respectively.



```
[[' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' 'X' ' ' ' ' ' ']
 [' ' ' ' ' ' 'O' 'X' ' ' ' ' ' ']
 [' ' ' ' 'O' 'X' 'O' ' ' ' ' ' ']
 [' ' 'O' 'O' 'O' 'X' ' ' ' ' 'X']]
```

Another part is the Monte-Carlo Tree Search: the Monte-Carlo Tree Search (MCTS) algorithm is devised to search in a smarter and more efficient way. Essentially, one wants to optimize the exploration-exploitation tradeoff, where one wants to search just exhaustively enough (exploration) to discover the best possible reward (exploitation).

Iteration Pipeline

In summary, a full iteration pipeline consists of:

1. Self-play using MCTS to generate game datasets (s, p, v), with the neural net guiding the search by providing the prior probabilities in choosing the action
2. Train the neural network using the (s, p, v) datasets generated from MCTS self-play

3. Evaluate the trained neural net (at predefined epoch checkpoints) by pitting it against the neural net from the previous iteration, again using MCTS guided by the respective neural nets, and keep only the neural net that performs better.
4. Rinse and repeat

Results

Over a period of several weeks of sporadic training on Google Colab, a total of 6 iterations for a total of 4902 MCTS self-play games was generated. I get the following results for the training:

Iteration 0:
alpha_net_0 (Initialized with random weights)
151 games of MCTS self-play generated

Iteration 1:
alpha_net_1 (trained from iteration 0)
148 games of MCTS self-play generated

Iteration 2:
alpha_net_2 (trained from iteration 1)
310 games of MCTS self-play generated

Evaluation 1:
Alpha_net_2 is pitted against alpha_net_0
Out of 100 games played, alpha_net_2 won 83 and lost 17

Iteration 3:
alpha_net_3 (trained from iteration 2)
584 games of MCTS self-play generated

Iteration 4:
alpha_net_4 (trained from iteration 3)
753 games of MCTS self-play generated

Iteration 5:
alpha_net_5 (trained from iteration 4)
1286 games of MCTS self-play generated

Iteration 6:
alpha_net_6 (trained from iteration 5)
1670 games of MCTS self-play generated

Evaluation 2:
Alpha_net_6 pitted against alpha_net_3
Out of 100 games played, alpha_net_6 won 92 and lost 8.

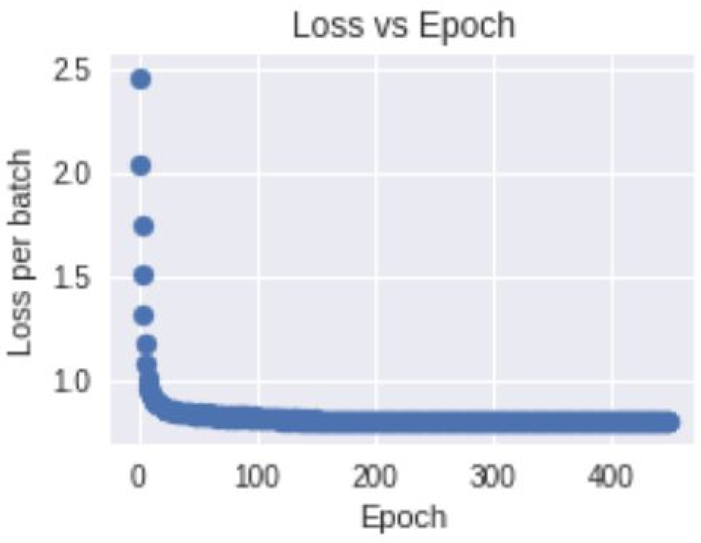


Figure 2. Loss vs Epochs

A loss vs epoch of the neural network training for each iteration is shown in figure 2, showing that training proceeds quite well. From both evaluations 1 & 2 at selected points in the iterations, we can see that the neural net is indeed always improving and becoming stronger than its previous version in generating winning moves.

References

1. Code retrained from following repository: https://github.com/kirarpit/connect4
2. My Github Repository Link: https://github.com/AkshithaPothamshetty/ENPM690HW4