Exercise 8: Implementing the PUCT Algorithm

January 16, 2025

In this project, you will implement the PUCT algorithm. This exercise will build on Exercises 4 and 7.

1. **Adapting MCTS for Your Board Game**
   - Adapt your MCTS code from Exercise 4 to your chosen board game from Exercise 7.
   - Debug thoroughly to ensure it plays the game correctly and performs well.
2. **Creating the Neural Network Class**
   - Write a Python class called `GameNetwork` for your neural network.
   - The network must have a value head estimating win probability and a policy head predicting action probabilities.
   - Ensure input and output dimensions match `encode` and `decode` from Exercise 7.
   - Include methods for saving and loading network weights.
3. **Implementing PUCT Algorithm**
   - Write `PUCTNode` and `PUCTPlayer` classes by adapting `MCTSNode` and `MCTSPlayer`.
   - Key changes:
     - Replace random rollouts with evaluations from the neural network.
     - Each node maintains a prior probability P from the policy head of its parent.
     - Each node maintains Q (average value) and N (visit count). Avoid calling it `wins`.
     - Use the PUCT scoring formula:

$$U(s,a) = Q(s,a) + c_{puct} \cdot P(s,a) \cdot \frac{\sqrt{N(s)}}{1 + N(s,a)}$$

4. **Pre-training the Neural Network**

   - Use your MCTSPlayer to generate 10,000 self-play games without network guidance.
   - Train the value head using game outcomes.
   - Train the policy head using visit counts at the root node.