SMART AI Automation Final Report

Author: Khushi Chaudhari

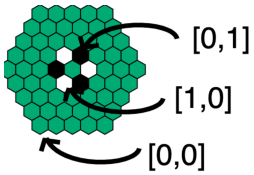
Overall Summary

The following report outlines an AI automation project. This involves training a set of AI agents, which are built by creating neural network brains to play the game of Othello. There is a small twist in Othella by placing it in a hexagonal board to give the game a unique twist, so the project is Hexathello. These AI agents are trained by playing against other AI agents and learning from past games over a numerous number of times. To explore the complexity of neural networks, the project varies the architecture in two ways: number of layers with fixed number of neurons and number of layers with fixed neurons per layer.

The Game of Othello

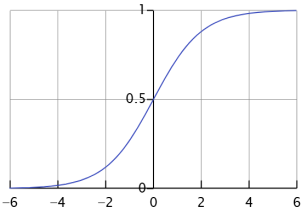
Othello is played on an 8x8 board with two players, one player playing black pieces and the other playing white pieces. At the start of the game, place two white and two black pieces at the center of the board, so the pairs of matching colors are diagonal from each other. Players take turns placing their colored discs on the board in which a placed disc must surround one or more of your opponent's discs in a straight line. This line can be horizontal, vertical, and even diagonal. When you surround one or more of your opponent's discs, the surrounded disc is flipped and turns to your color. The game continues until neither player can make a legal move. At that point, the player with the most discs on the board wins.

Encoding the Board

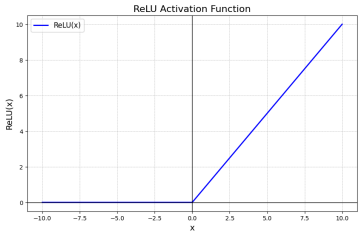


By using hexagonal grids as a playing board, we can represent the position of a spot using coordinate tuples. This example shows a 5x5 board, where 5 comes from the length of one side. The coordinates on the right illustrate a few spots and indicate if they are occupied, and if so, by which color.

Neural Networks and Functions Used



Much like the human brain, a neural network consists of interconnected units called neurons. These neurons are organized into a layer, and a neural network typically has many layers. In our case, each of these neurons takes an input, the encoded board state, and processes it using knowledge gained from prior training.



This is done by using an activation function to determine the most appropriate action. For example, we utilized the sigmoid function, which maps input values from negative to positive infinity into a range between zero and one. This helps the network decide which neurons to activate, ultimately guiding the agent's next move. Below is what the sigmoid and relu functions look like, both of which were used to develop our agents and their brains.

Training the Agents

We used baseline models to play Hexathello games against each other. The baseline models trained following the greedy strategy, picking the move that would be best for the given board. So, these greedy models did not use any long-term strategy. Each game generated a

sequence of moves and a winner, which was then stored as a history. The agents then trained on the histories before playing more games against each other. They used the training data to decide on moves. There was also an added randomness in each agent so that they would make some moves that weren't necessarily predictable. This was to make sure the agents did not continue to play the exact same moves. Agents were then able to learn from themselves and become smarter by simply playing more games.

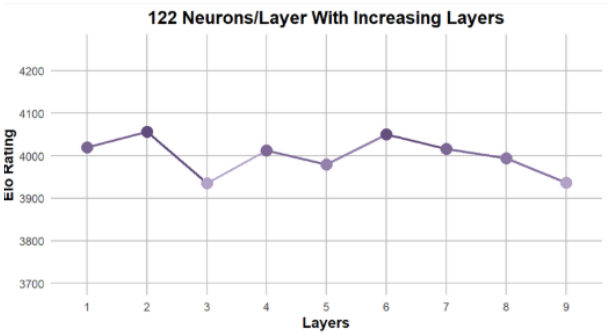
The Experiment

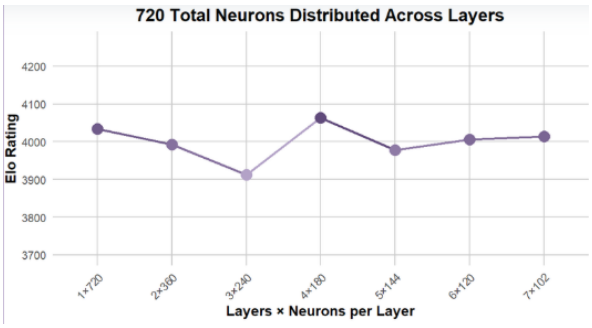
It is widely accepted that more complex neural networks, those with more layers, are better suited for identifying and learning complex patterns in data. To explore this idea, we trained and tested our AI agents by having them compete against each other. Every agent started at a skill rating (elo) of 4000. Every win, loss, and tie was calculated using the chess elo rating system. We manipulated the complexity of each neural network in two ways:

1. Distributing a fixed total number of 720 neurons across an increasing number of layers, starting at one and ending at seven layers.
2. Increasing the number of layers, starting from one and ending at nine, across nine agents while keeping the number of neurons per layer constant.

Conclusion

In both experiments, the agents that achieved the highest elo ratings were those utilizing a moderate number of layers, somewhere in the middle of the lowest and highest number of layers. These mid-range agents seemed to achieve an effective balance between model complexity and generalization. Specifically, they possessed sufficient depth to capture unique and intricate patterns in strategic decision-making, while avoiding excessive complexity that could lead to overfitting.





In contrast, it is possible that agents with too few layers lacked the complexity to learn nuanced behaviors, and those with too many layers may have overfit, reducing their adaptability across diverse board states. This suggests that there exists an optimal range of network depth that maximizes both learning efficiency and competitive performance in this domain. While deeper networks with more layers are theoretically capable of learning more complex patterns, more layers do not necessarily equate to better performance in our case.

Therefore, it is crucial to understand the desired behaviors and objectives for the agents in order to design a model that strikes the right balance. That is, finding the "sweet spot" in network complexity that enhances both learning capacity and playstyle diversity at the same time.