MATH 302 SPRING 2022
TERM PROJECT

Topic: Implementing a Neural Network to Play Hex

Group Number: 18

Duy Nguyen (301379431)
Khushwant Parmar (301370994)
Stephen Ng (301352700)

Introduction

Hex is a two-player game in which the players must place tiles to form a monochromatic line across a board to connect one side to the opposite side. Each player can only place one tile during their turn with black usually moving first, and no move may be repeated. While there are many board sizes, each board must be of size *n* x *n* with the most common boards being the 11x11 or the 13x13 which offer complex enough play while not favoring either player. The inherent design of Hex also prevents draws from occurring, meaning one player will always win, as proven by David Gale using the Brouwer fixed-point theorem (Gale, 1979). But while there is always a winner, there are no strategies that guarantee a certain player wins. As well, for larger, standard boards, an exhaustive search for all openings and strategies is next to impossible, even for computational programs. And “solving arbitrary Hex positions is PSPACE-complete”, this is more difficult than NP-complete, which is utilizing a brute-force search algorithm (Gao, 2022).

While Hex has been available for a few decades, it was only recent developments in computational programs that has allowed computers to play against humans. Early Hex-playing models used methods similar to chess-playing programs, that is, deep selective minimax search, but most found that this method was not as effective when applied to games of Hex (Gao et al., 2017). It was not until Vadim Anshelevich’s work on Hex programs that the first modern Hex-playing programs were developed with Hexy, the first Hex-playing program to win the Computer Olympiad in 2000. But with this advancement, there became an arms race of sorts to see how strong the programs could become.

Six was the next iteration of Hex-playing programs taking gold in the 8th Olympiad, just three years after arriving on the scene. It held the title until losing to Wolve in the 13th Olympiad, a reign of six years. Wolve held the title for just one year however, losing it to MoHex. And it was these two programs, Wolve and MoHex, that constantly fought for supremacy, taking first and second places in each successive Olympiad against each other, a rivalry of sorts. Each of their iterations is stronger than the last, with even just MoHex 2.0 being about 250 points stronger than MoHex on the 11x11 board and 300 on the 13x13 board. The next advancement in Hex-playing programs has been the implementation of Deep Neural Networks used to solve Hex openings. Using these neural network models instead of previous models, opening moves were made 2.6-4.1 times faster depending on the model (Gao, 2020).

Hex-Playing Programs: How They Tick

The term Deep Neural Network refers to Artificial Neural Networks (ANN) with multiple layers.(Albawi,2018) Convolutional Neural Network is one of the most popular neural networks named after the mathematical linear operation between matrices called convolution.(Albawi, 2018)

A typical CNN takes images as input, assigns importance (weights and biases) to the various aspects of the image, and differentiate them one from one another. (Saha, 2018) Its main objective is to reduce images into a form which is easier to process without losing important information. (Saha, 2018) CNNs have multiple layers, this includes a convolutional layer, non-linearity layer, pooling layer and fully-connected layer. (Saha, 2018)

The first layer's operation being Convolution is to extract the high-level features such as edges, from the input image. (Saha, 2018) It captures the Low-Level features such as edges, color, gradient orientation, etc. To further improve the performance of the convolutional layer more layers can be added which give us a network that has a wholesome understanding of images in the dataset. (Saha, 2018)

The motivation to use Neural Networks for Hex

As with another board game, Go, Hex has an extremely large set of moves to play. This large number of moves makes traditional programs that utilize fixed-depth exhaustive search inefficient due to the computational time required to find a set of moves. However, one key difference between Go and Hex is that Hex has a "strong evaluation function" which allows more specified search programs that implement "knowledge and connection strategies" to be more successful in finding optimal opening moves (Gao et al., 2022).

Recent developments in computer Go by Google Deep Technologies has shown the superiority of using Convolutional Neural Networks for representation and learning game knowledge. Motivated by the success in computer Go, deep convolutional neural nets were applied to represent and learn knowledge in Hex. The ultimate goal in computer Hex is to get to the level of an AlphaGo style playing system for computer Hex (Gao et al., 2022).

Approach

We examine the applicability of a Deep Convolutional Neural Network Hex 13x13 agent without the use of Monte Carlo Tree Search. We trained a move prediction Deep-CNN model (Gao, 2017) using a dataset of human-played games scraped from LittleGolem (Frederick, 2018). Our model predicts the best move based on the current board states to make a play. We ignore swap rules and will be examining the win rate as black and as white separately in later sections.

Data

Our dataset is collected from a dataset of 45446 human-played games on 13x13 board size scraped from LittleGolem by (Frederick, 2018).

The dataset includes a wide variety of skill levels, with an average site skill rating of 1731.62. The distribution of skill levels can be seen in Figure 1.

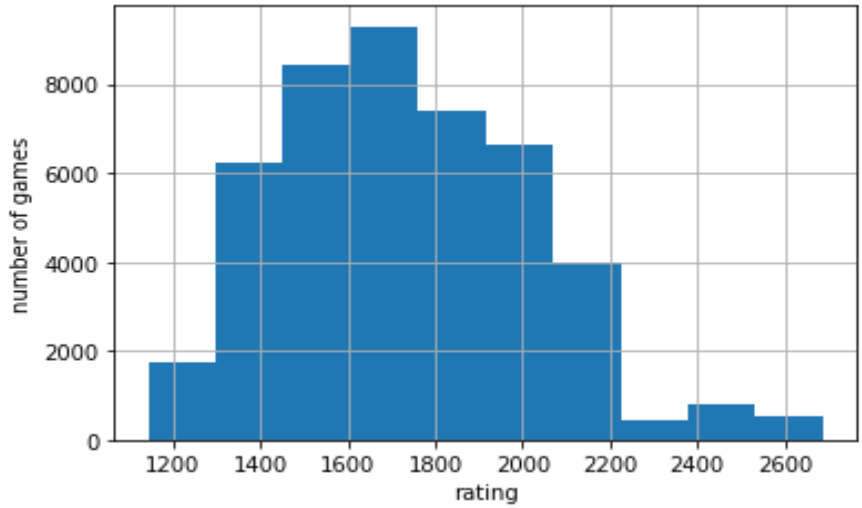

Figure 1: Distribution of user skill ratings

We notice the dataset is positively skewed, and thus the majority of games are of lower skills. Due to the lack of high-level games, we decided to keep all data for training.

We preprocess the data by removing games with less than 2 moves and excluding games using the swap rule, leaving us with 23475 games. We then extract 808457 distinct state-action pairs to make our dataset.

Input Processing

We preprocess each state-action pair into feature planes and one-hot vectors before feeding them into our model.

For each state, we represent them as two 17x17 input planes representing the moves each player has made. We included additional border padding to represent the game's objective of connecting 2 sides of the board.

For each action, we represent them as a 169 dimension one-hot vector, with each dimension corresponding to a move on the board.

Architecture and Training

Our architecture closely resembles that of Gao Chan (Gao, 2017), with a 5x5 convolution layer stacked with 4 3x3 convolution layers. Each convolution layer has 128 filters and a stride value of 1. A final convolution layer is a single filter of size 1x1 with stride 1. The last layer is a dense layer with softmax activation for classification. This amounts to a total of 646,003 trainable parameters. The output of the model is a probability distribution of all moves on the board.

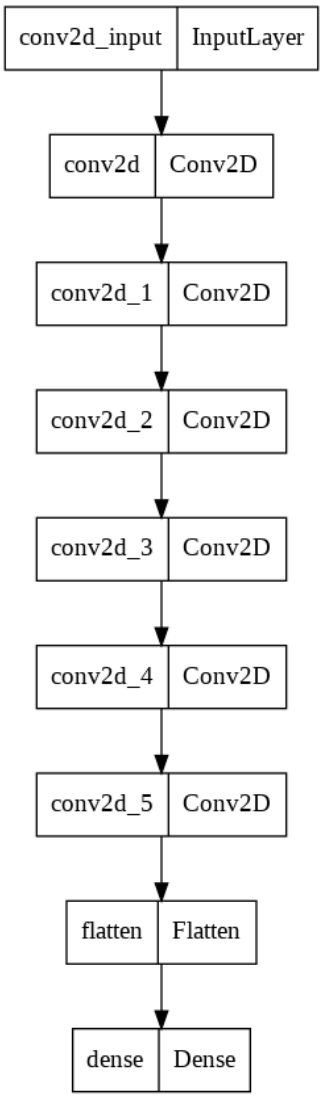

Figure 2: Model Architecture

We use Categorical Cross-Entropy for the loss function and Adam gradient descent optimizer. We train the model for 20 epochs with batch size of 128 for a total of 113700 gradient update steps. We train using 90% of our data, with 10% resolved as the test set. The loss and accuracy of training set for collected every epoch. The training took 40 minutes on a Google Collab instance.

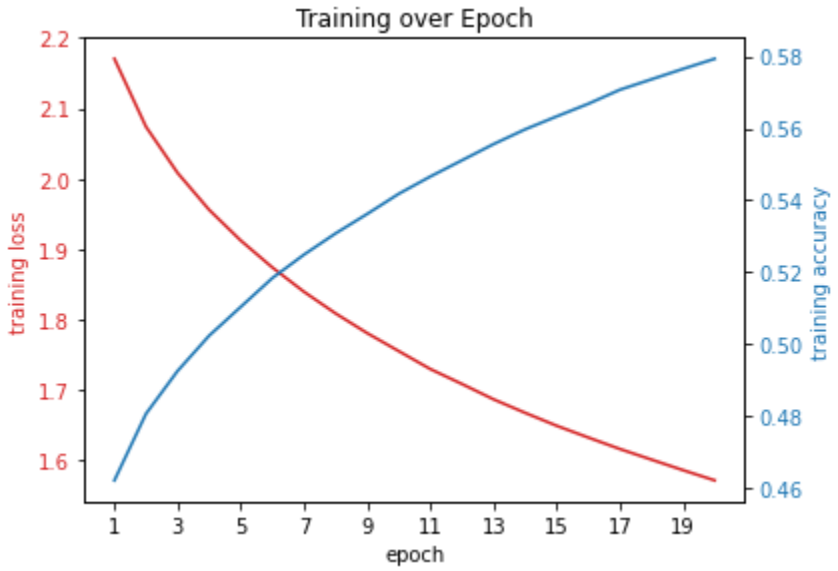

Figure 3: Training loss and accuracy over epochs

After 20 epochs, the model was able to accurately predict 41.69% of our test set.

Evaluation

We evaluate our model's performance against a random player using win rates over 200 simulated games. We also played 2 games against MoHex 2.0 (Gao, 2020). However, due to MoHex 2.0 being implemented in a different language, we were unable to simulate large amounts of simulated games against it.

Since we don't allow swap rules, playing as Black has a significant advantage.

Simulated Game Data

Opponent	As white	As Black	Combined
Random Player	44%	65%	54.5%

Our data shows a relatively low win rate than expected against just a naive player picking random moves. Overall, the model still outperformed a random player, but by a smaller margin than we hoped for.

One possible explanation we could come up with is that our CNN model overfitted the training data, and thus couldn't account for a player with no rationale behind their moves. Since the pattern of play of a random player is very different from a normal

human player, our model might have been unable to react to this type of play and ended up playing random moves. Most games simulated last for very long with most of the board filled, which supports this hypothesis.

Against MoHex 2.0

Our model played 2 games against MoHex 2.0 (Gao, 2020) as the Black and the White player. The final board position can be seen in our figure.

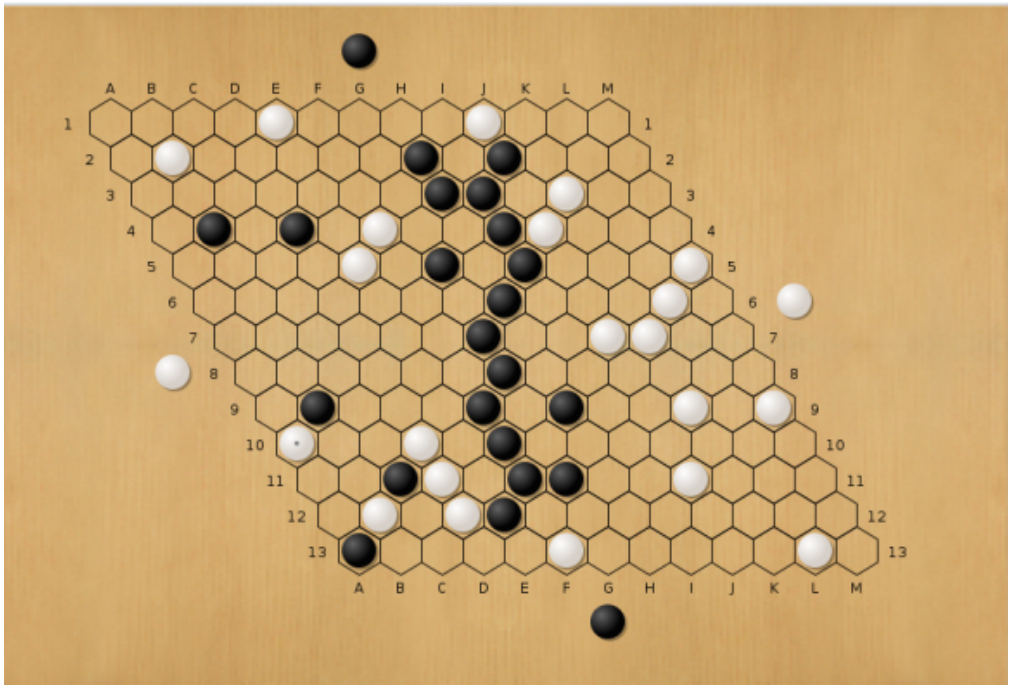

Figure: Our CNN model (White) vs MoHex 2.0 (Black)

In the first game playing as White, our model made some important defenses against MoHex 2.0's (Black) chains, such as 1J, 4J, 12D, 12B, 13F. Offensively, the model tried to build up a chain from 5M to 7J. However, other moves from the model are not explainable, such as 1E, 2B, etc. which ultimately wasted too many moves to successfully defend against MoHex 2.0.

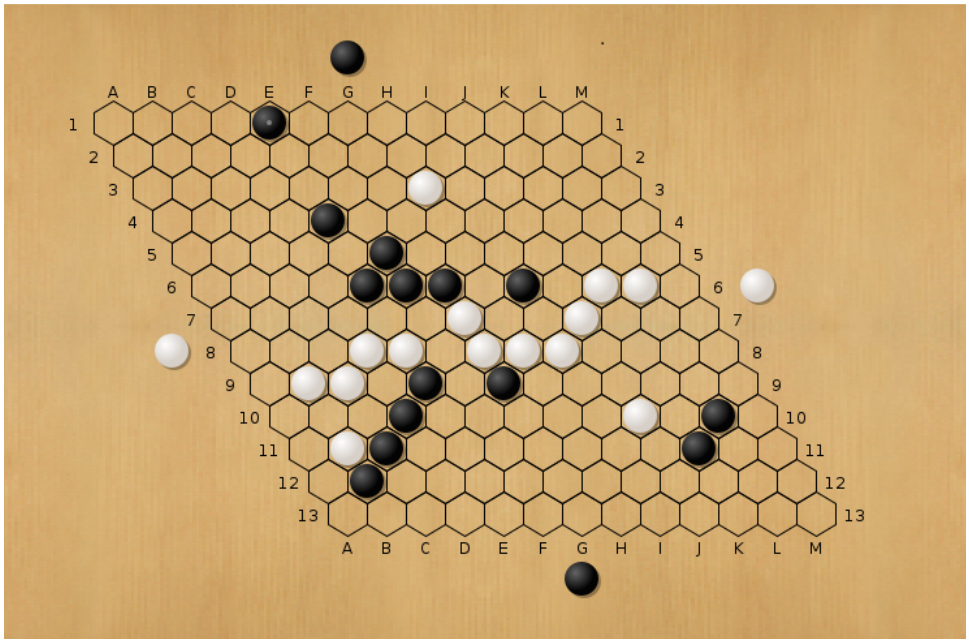

Figure: Our CNN model (Black) vs MoHex 2.0 (White)

In the second game playing as Black, the model showed clear offensive tendency and had a long chain in 12B-9E that could connect to 6F and more. However, similar to the last game, the model still had unexplainable moves such as 10L and 11K that threw away its advantage of playing as Black and ultimately lost to MoHex 2.0.

The Monte Carlo Tree Search Component of MoHex 2.0 allows it to plan several moves ahead of our model, thus we believe the loss of search makes for a greedy model that will not be able to compete in a complex game as Hex.

Conclusion

Our pure Deep-CNN model performed better than a pure random agent, but was unable to compete against a competent Monte Carlo search-based program MoHex. We conclude that the benefits of a neural network (CNN in particular) doesn't outweigh the benefit of classical search.

Additionally, through our evaluation, we identifies possible weak points in the research:

- Dataset: Due to the lack of quantity of high-level human plays in Hex compared to more popular games such as Chess, our model didn't have enough good examples to learn from..
- Overfitting: The model wasn't good at generalization, as evident by a relatively low win rate against a random opponent and a low accuracy in test set.

To alleviate these weak points, our future researchers should focus on:

- Using self-play data from state-of-the-art AI models which outperformed human players.
- Using drop-off to avoid overfitting to the training data.

References

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2018, March 8). *Understanding of a convolutional neural network*. IEEE Xplore. Retrieved April 6, 2022, from https://ieeexplore.ieee.org/abstract/document/8308186

Chess Programming Wiki. (2020, August 5). *5th Computer olympiad*. 5th Computer Olympiad - Chessprogramming wiki. Retrieved April 6, 2022, from https://www.chessprogramming.org/5th_Computer_Olympiad

Frederick, J., & Egbert, H. (2018). *HI4A4/hex-ai: A neural network to play the game hex: Http://www.lutanho.net/play/hex.html*. GitHub. Retrieved April 6, 2022, from https://github.com/hi4a4/Hex-AI

Gale, D. (1978). *The game of hex and the Brouwer fixed point theorem*. Taylor & Francis, Ltd.

Gao, C. (2020). *CGAO3/benzene-vanilla-cmake: Cmake managed benzene vanilla for playing and solving the game of hex, easier to install!* GitHub. Retrieved April 6, 2022, from https://github.com/cgao3/benzene-vanilla-cmake

Gao, C. (2020). *Search and Learning Algorithms for Two-Player Games with Application to the Game of Hex* (thesis).

Gao, C. (2022). *Complexity aspects of Hex*. Chao Gao / home. Retrieved April 6, 2022, from https://cgao3.github.io/files/WhyHex/

Gao, C., & Hayward, R. (2017). *Move prediction using deep ... - university of Alberta*. IEEE. Retrieved April 7, 2022, from https://webdocs.cs.ualberta.ca/~hayward/papers/movepredhex.pdf

Gao, C., Müller, M., & Hayward, R. (2017). Focused depth-first proof number search using convolutional neural networks for the game of hex. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. https://doi.org/10.24963/ijcai.2017/513

Huang, S.-C., Arneson, B., Hayward, R. B., Müller, M., & Pawlewicz, J. (2014). MoHex 2.0: A pattern-based MCTS hex player. *Computers and Games*, 60–71. https://doi.org/10.1007/978-3-319-09165-5_6

Saha, S. (2018, December 17). *A comprehensive guide to Convolutional Neural Networks - the eli5 way*. Medium. Retrieved April 6, 2022, from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53