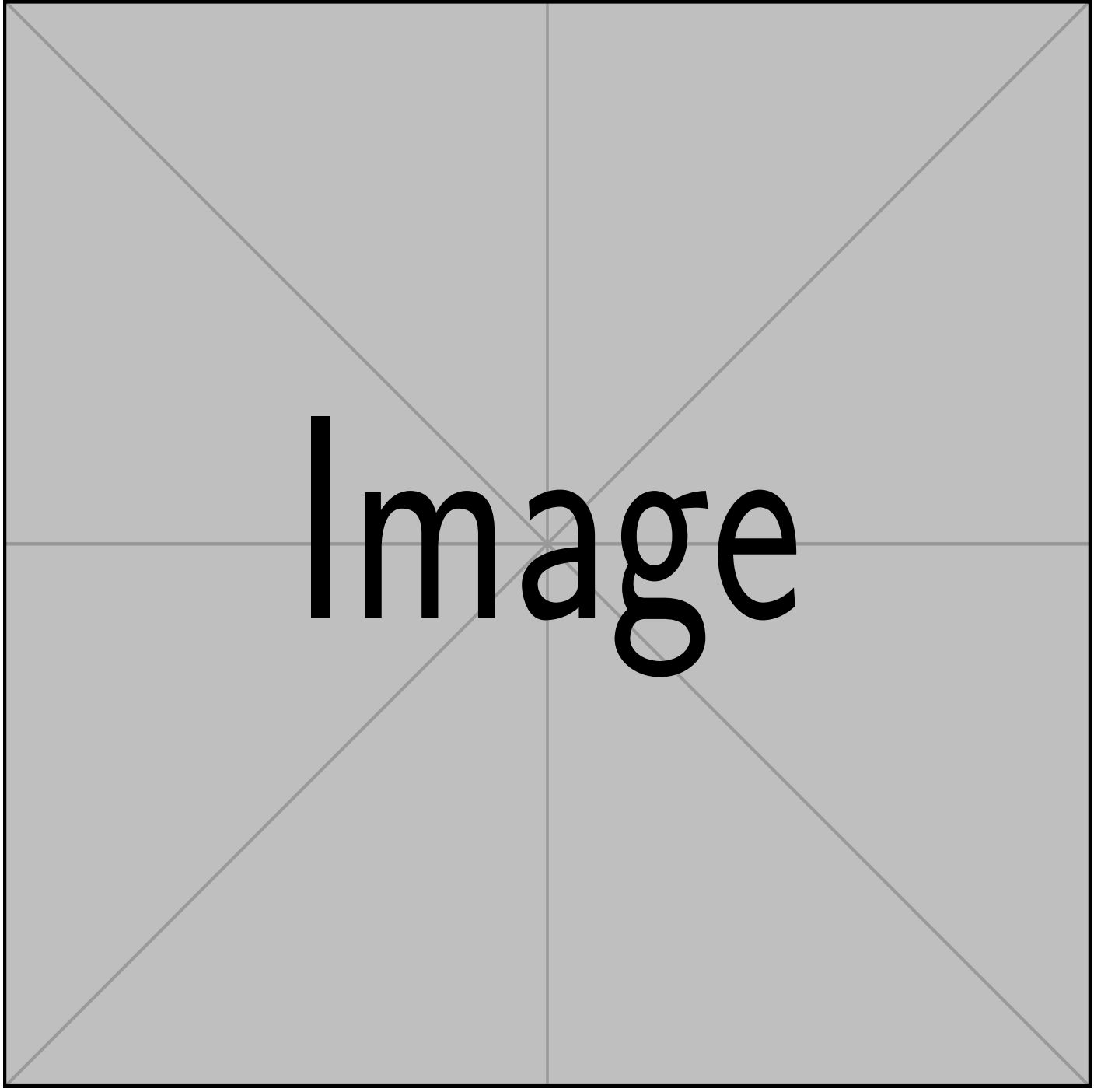

HOW TO CREATE A CHESS ENGINE USING DEEP REINFORCEMENT LEARNING

A CRITICAL LOOK AT DEEPMIND'S ALPHAZERO

INTERAL PROMOTOR: WOUTER GEVAERT

EXTERNAL PROMOTOR: <NAME HERE>

RESEARCH CONDUCTED BY

TUUR VANHOUTTE

FOR OBTAINING A BACHELOR'S DEGREE IN

MULTIMEDIA & CREATIVE TECHNOLOGIES

HOWEST | 2021-2022

Preface

This bachelor thesis is the conclusion to the bachelor programme Multimedia & Creative Technologies at Howest college West Flanders in Kortrijk, Belgium. The programme teaches students a wide range of skills in the field of computer science, with a focus on creativity and Internet of Things. From the second year on, students can choose between four different modules:

1. **AI Engineer**
2. **Smart XR Developer**
3. **Next Web Developer**
4. **IoT Infrastructure Engineer**

This bachelor thesis was made under the **AI Engineer** module. The subject of the thesis is a critical look at the result of my research project in the previous semester. The goal of the project was to create a chess engine in Python with deep reinforcement learning based on DeepMind's AlphaZero algorithm.

I will explain the research I needed to create it, the technical details on how to program the chess engine and I will reflect on the results of the project. To do this, I will contact multiple people familiar with the field of reinforcement learning to get a better understanding of the impact of this research on society. Based on this, I will give advice to people and companies who wish to implement similar algorithms.

I would like to show gratitude to Wouter Gevaert for his enthousiastic support in the creation of my research project and this thesis. I also want to thank the other teachers at Howest Kortrijk, who shared their knowledge and expertise in programming and AI in very interesting classes.

Furthermore, I would liike to thank my parents for giving me the chance to have a good education, and the motivation to get the best I can out of my studies.

Tuur Vanhoutte, 1st June 2022

Abstract

This bachelor thesis answers the question: “How to create a chess engine using deep reinforcement learning?”. It explains the difference between normal chess engines and chess engines that use deep reinforcement learning, and specifically tries to recreate the results of AlphaZero, the chess engine by DeepMind, in Python on consumer hardware.

The technical research shows what is needed to create my implementation using Python and TensorFlow. It shows how to program the chess engine, how to build the neural network, and how to train and evaluate the network. During the creation of this chess engine, it was crucial to create a huge amount of data through self-play.

The thesis contains a reflection on the results of my research project, which proposes a solution to the problem of creating a high amount of games through self-play. It also reflects on the impact of this research on society, and the viability of this type of artificial intelligence in the future. With this comes a section on advice for companies that wish to implement similar algorithms.

Contents

List of figures

List of abbreviations

Glossary

1 Introduction

Chess is not only one of the most popular board games in the world, it is also a breeding ground for complex algorithms and more recently, machine learning. Chess is theoretically a deterministic game: no information is hidden from either player and every position has a calculable set of possible moves. Because the branching factor of chess is about 35-38 moves [1], calculating if a position is winning or losing requires an enormous amount of calculations.

Throughout the entire history of computer science, researchers have continously tried to find better ways to calculate if a position is winning or losing. The most famous example is the StockFish engine [2], which uses the minimax algorithm with alpha-beta pruning to calculate the best move.

More recently, researchers at Google DeepMind have developed a new algorithm called AlphaZero [3]. This thesis explores the concept of AlphaZero, how to create a chess engine based on it, and the impact of the algorithm on both the world of chess and the rest of society.

Research has been conducted by investigating what is needed to recreate the results of AlphaZero, by programming a simple implementation using Python and TensorFlow. This was done as part of a research project between November 2021 and January 2022. The code was written with lots of trial and error, as DeepMind released very little information about the detailed workings of the algorithm.

2 Research

2.1 What is a chess engine?

According to Wikipedia [4], a chess engine is a computer program that analyzes chess or chess variant positions, and generates a move or list of moves that it regards as strongest. Given any chess position, the engine will estimate the winner of that position based on the strength of the possible future moves up to a certain depth. The strength of a chess engine is determined by the amount of moves, both in depth and breadth, that the engine can calcuate.

2.2 How do traditional chess engines work?

Contemporary chess engines, like Stockfish, use a variant of the minimax algorithm that employs alpha-beta pruning.

2.2.1 The minimax algorithm

The minimax algorithm [5] is a general algorithm usable in many applications, ranging from artificial intelligence to decision theory and game theory. The algorithm tries to minimize the maximum amount of loss. In chess, this means that the engine tries to minimize the possibility for the worst-case scenario: the opponent checkmating the player. For games where the player needs to maximize a score, the algorithm is called maximin: maximizing the minimum gain.

Minimax creates a tree with chess positions as nodes and chess moves as edges between the nodes. Each node gets a value that represents the strength of the position for the current player. It starts with only the root of the tree as the current position. The algorithm explores the tree in a depth-first manner. This means that it will traverse the tree vertically until a certain depth is reached. When that happens, the algorithm evaluates that node's position and returns its value to its parent node.

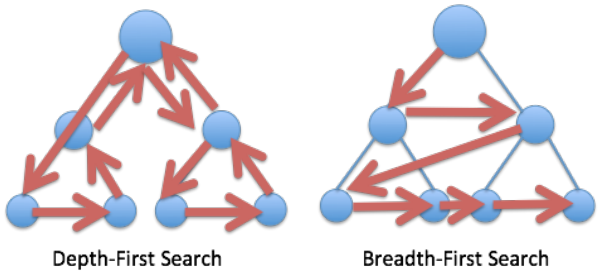

Figure 1: Depth-First search vs Breadth-First search [6]

2.2.2 Pseudocode

The algorithm is recursive: it calls itself with different arguments, depending on which player's turn it is. In chess, white wants to maximize the score, and black wants to minimize it. [5]

```
function  minimax(node, depth, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := - inf
        for each child of node do
            value := max(value, minimax(child, depth - 1, FALSE))
        return value
    else (* minimizing player *)
        value := + inf
```

```
        for each child of node do
            value := min(value, minimax(child, depth - 1, TRUE))
        return value
```

Calling the function:

```
minimax(origin, depth, TRUE)
```

2.2.3 Alpha-beta pruning

Alpha-beta pruning is an algorithm [7] to reduce the amount of nodes that need to be evaluated by minimax. It does this by cutting away tree branches that lead to worse outcomes.

2.3 AlphaZero

3 Technical research

4 Reflection

5 Advice

6 Conclusion

7 Bibliography

[1] *Branching Factor - Chessprogramming wiki*. [Online]. Available: https://www.chessprogramming.org/Branching_Factor (visited on 03/28/2022).

[2] "Stockfish (chess)," *Wikipedia*, Mar. 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Stockfish_(chess)&oldid=1079146589 (visited on 03/28/2022).

[3] "AlphaZero," *Wikipedia*, Jan. 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=AlphaZero&oldid=1065791194 (visited on 02/01/2022).

[4] "Chess engine," *Wikipedia*, Apr. 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Chess_engine&oldid=1080874516 (visited on 04/05/2022).

[5] "Minimax," *Wikipedia*, Mar. 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Minimax&oldid=1076761456 (visited on 04/05/2022).

[6] M. Eppes, *How a Computerized Chess Opponent "Thinks" — The Minimax Algorithm*, Oct. 2019. [Online]. Available: https://towardsdatascience.com/how-a-chess-playing-computer-thinks-about-its-next-move-8f028bd0e7b1 (visited on 04/05/2022).

[7] "Alpha–beta pruning," *Wikipedia*, Jan. 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Alpha%E2%80%93beta_pruning&oldid=1068746141 (visited on 02/01/2022).

8 Appendix