# Analyzing MuZero and implementing it for simpler games

Ashwin Ramdas
Data Analytics
Dublin Business School
Dublin, Leinster, Ireland
10534162@dbs.ie

## ABSTRACT

Creating simulations and finding every possibility for strategy games has been an area of research for a long time. Before the advent into unsupervised learning by the DeepMind team, engines for most games like chess have used supervised learning to find better moves for players and beating them, starting with the second match of IBM Deep Blue against then world champion Garry Kasparov, which the engine won comprehensively in regulation time. However, MuZero and its predecessors have been a revelation using Deep Neural Networks to implement unsupervised learning to create such engines. This research proposal contains a brief history of chess engines and then its advent into other strategy games as well, namely, Go, checkers, poker etc. It then talks about our motivations, which is followed by the Research Question and the purpose and variables needed for this research. The next part consists of the Literature Review of the topic and method. The method in this topic is same as the ground breaking paper by the DeepMind team titled "Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model" which tells us about the implementation of the engine which we aim to use to for a simpler game like Tetris.

## CCS CONCEPTS

• Reinforcement Learning • Supervised learning and Unsupervised learning

## KEYWORDS

Alpha Go, AlphaGo Zero , MuZero, self-play

# 1. Introduction

## 1.1. Project Background and Motivation

The foray into the creation of chess engines started in 1970-80's and there were matches that were won by different sets off machines against different players. However, the spark that started the fire was with IBM Deep Blue which beat then World Champion Gary Kasparov in 1997, in regulation time for the first time ever. Since then, there have been a huge number of chess engines which all used supervised learning to take databases of all the official chess games played and, then, give any player the next best move and predict the outcomes of any match being played. However, the DeepMind team developed an algorithm to make the engine learn chess by itself, which was a revolution. However, before they came to chess, they mastered a more complex game called Go, revolutionizing the way we think about the strategies in Go.

With the introduction of Alpha Go, the world saw a revolution of sorts as a game as complex as Go, which has about 200 permutation per move, was mastered by a computer. It started with the 5-0 defeat of 2016 European Championship winner Fan Hui. Then, after about six months, the 4-1 defeat of World Champion Lee Sedol in Seoul, where in game one, the machine showed a move never before seen in the Go world. So, not only was this machine revolutionary in the development of AI, it also helped us understand the game of Go better.

The rules of Go states that the player who has the most area of the board wins. Therefore, the classical approach was to grab as much as area as possible, as soon as possible. But Alpha Go Lee, as it came to be known as after the defeat of the 18-time world champion, showed in the last game that a win is a win irrespective of the margin, as it went on to win by two and half points.

Lee Sedol won every game for 2 months after the match with Alpha Go Lee, showing he also ended up learning a lot from it.

After this monumental game, the DeepMind tackled the complexities of chess with the new Alpha Go Zero, which after learning the game by itself for four hours, won 28 of its 100 games against StockFish, generally considered the best Chess engine before Alpha Go Zero, while the StockFish won zero.

The DeepMind team then, went on to tackle games like Shogi, Atari etc. This led to many engines by that team, each playing a different game. And as recently as November 2019, the team created MuZero another milestone in AI development.

To understand why it was a milestone, we have to understand how Alpha Zero worked. The machine starts learning 'tabula rasa' meaning with a blank slate. As it starts running, the Alpha Zero system is given a set of rules and a playing environment of which ever game it was playing.

The main difference between Alpha Zero and MuZero is the fact that MuZero is not even given the rules of the game, but only given a game state and as it plays it is told that what it played was a legal move or not. It is not even told how the winner will be decided, only when it wins a match, does it get to know. It is sort of like a child learns the rule of life. It has since its inception mastered games like chess, go, shogi, checkers, poker, all Atari developmental environmental games, StarCraft 2 (ranking better than 98.8% of players in that game, with a name as cool as AlphaStar).

### 1.2. Research Question

The research question for this study is "How is MuZero so good at games like chess & go and can it be applied to other similar games? Investigation into existing techniques."

### 1.3. Purpose

The research question aims to understand the inner workings of an unsupervised learning engine as complex as MuZero and implement it for similar simpler games like Tetris using the algorithm and pseudo code provided in the MuZero paper.

### 1.4. Research Variables

The research variables to be considered in this project is the game itself as MuZero is an unsupervised learning algorithm that takes game state as an input. For each game, there is different number of input that it will process. However, for each move in any game, the engine saves the variables:

a) The game state
b) The search probabilities
c) The winner

Where each will be processed using the Monte-Carlo Tree Search (MCTS)

# 2. Literature Review

There has been a huge amount of research done in the field of Chess engines and how it performs. However, almost none of them used Unsupervised learning to take decisions. So, to check related works to MuZero, we would have to check the predecessors of MuZero itself, namely AlphaGo Zero.

This section contains information about the works by AlphaGo Zero and other literature reviews about the methods that have been used as well as the method which is going to be implemented.

## 2.1. Related Work.

The section has reviews about the works that have already been completed including AlphaGo Zero, IBM Deep blue etc. Starting with Deep Blue, it worked on 480 custom chess chips made at that time, which was fed data about the previous chess games in history and it analysed each game, to take the better decisions during any new game.

AlphaGo Zero on the other hand is an engine that uses unsupervised learning to learn the game and then, used Monte Carlo Tree Search (MCTS) to find the right move.

AlphaGo Zero is has dual- res layer network, which means that the first layer block is a combination of value head and policy head. This is followed by about 40 residual blocks, finally ending with a convolutional layer block. The MCTS stores N,W,Q and P at each node, where N is the number of times action a has been taken for state s, W is the total value of the next state, Q is the mean value of the next state and P is the prior probability of selecting action a.

## 2.2. Review about the MuZero research.

MuZero is very similar to AlphaGo Zero in terms of structure. However, where it fundamentally differs from it is the fact that any game it plays, the rules are not told to it beforehand and as soon as it makes a move, it is told if that was legal or not. Moreover, it is not told what its goal in the game is. Therefore, it has no idea what to attack to win and when it does end up winning, is when it is told that that was a win.

## 2.3. Review about the Methods.

As MuZero's working are almost exactly similar to that of AlphaGo Zero, the methods used will be almost exactly same as AlphaGo Zero.

# 3. Research Method and Specification

## 3.1. Collecting required data.

First and foremost, the data that needs to be collected is the program file for the game that needs to be played. As it mostly plays its own games, no additional data is to be collected as such. However, when we need to test the data, we need some data of previously played games to be used as test data. So, we say that the program doesn't need input data, however, it requires data to check its performance and all the test data needed will be taken from open sources with due reference given to the rightful owner.

## 3.2. Data Cleaning and Transformation

There is no input data, therefore, there is no need for data cleaning or transformation. However, the test data may need to cleaned and transformed according to the game state that needs to be input in the model.

## 3.3. Data Modelling

It uses a Deep neural network to operate which has a network of convolutional blocks with residual blocks with a dual policy and value block at the start. The number of layers/blocks vary with the type of game that is being played. Although in general, It has more blocks for more intensive games like StarCraft 2 and all the 57 Atari Development environment games.

## 3.4. Simulation

It learns with 'tabula rasa' (i.e. with a blank slate). Unlike Alpha Go Zero, it is not even fed the rules of the game. Therefore, it has to figure out the rules as well as the condition which is considered a win, as it plays the game with itself, millions of times. Then, it retains the optimal network which it chooses based on cross-entropy, mean squared error and regularization. It, then, makes the latest and current best neural network play about 500 games and the latest player has to win 55% of matches to become current best network.

After the training, the best network is then pitted against other opponents than itself. It is said to be in game state. Depending on what game it is playing, it takes a set of input in game state and finds the best move.

To find the best move, it does a Monte Carlo Tree Search (MCTS) to decide on the best move.

The DeepMind team has published a pseudo code along with the nitty-gritty details of the project in the Nature magazine. My aim is to use the article to write a code for a simpler game.

In my code, I'll also implement a similar architecture albeit simpler, with dual value and policy block at the start and residual blocks after that. At the end there will be a convolutional block.

It will also start with implementing the game environment after which the game will be played by the network but the illegal moves and the end of any game, i.e. win, loss or draw, will be coded to be provided as the network reaches there. In each move of self-play, the network will store the game state, the search probability and the winner. This will help later when it has to search through millions of permutations. After which the network weights will be optimized, by choosing the current best network based on cross-entropy, mean squared error and regularization.

After every 1000 training loop, the current best network will be pitted against the latest network and whoever, has 55% win rate will become the next current best network.

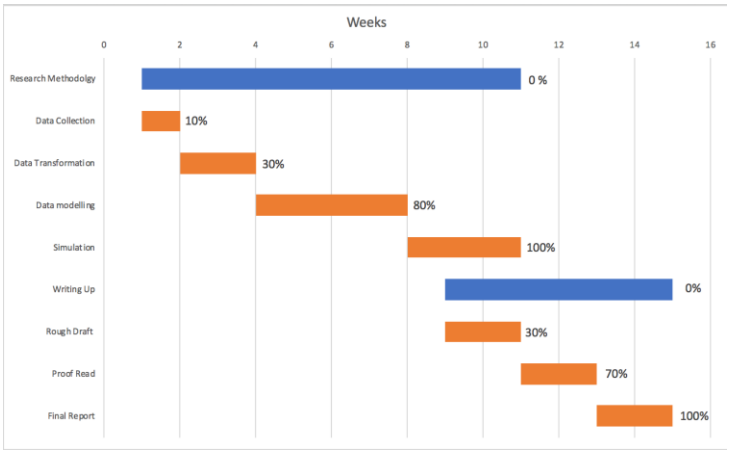
After all the training is complete the network will be pitted against me and my friends in the respective game and we'll see how well it performs.

The algorithm will then use MCTS to find the best move where at each MCTS node, it will store four numbers N,W,Q and P at each node, where N is the number of times action a has been taken for state s, W is the total value of the next state, Q is the mean value of the next state and P is the prior probability of selecting action a, which exactly same as AlphaGo Zero.

# 4. Ethical Considerations and Project Plan

As it is not an original project, but an investigation into existing methods, the ethical considerations that needs to be taken include is the due recognition that is to be given to the wonderful DeepMind team and its head Demis Hassabis.

A weekly plan of which part of the project to be completed has been plotted in a Gantt Chart. Figure 3 shows the an overview of when each process would be completed as per the guidelines for the completion of the project




## REFERENCES


[1] Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T. and Lillicrap, T., 2019. Mastering atari, go, chess and shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265*.

[2] Campbell, M., Hoane Jr, A.J. and Hsu, F.H., 2002. Deep blue. *Artificial intelligence*, *134*(1-2), pp.57-83.

[3] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A. and Chen, Y., 2017. Mastering the game of go without human knowledge. *Nature*, *550*(7676), pp.354-359.