Analyzing MuZero and implementing it for simpler games

Ashwin Ramdas

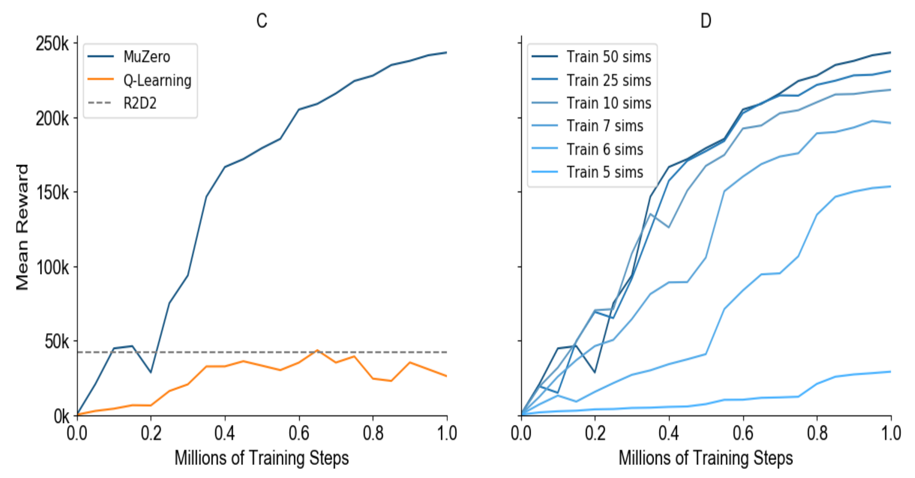
Introduction.

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.

MuZero and its predecessors

- **> Learns through self-play**
- **> Unsupervised Learning implementation**
- **> Has beaten World Champions in almost every game it played**
- **> Has Dual-res** Neural Network
- **> A Dual value and policy head block**
- **> About 40 residual blocks**
- **> A convolutional layer block at the end**

Performance on Pacman

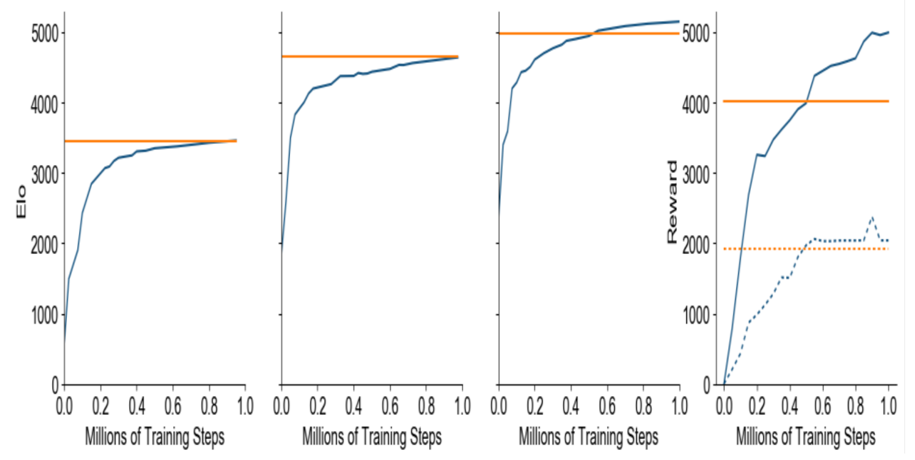


Evaluations of MuZero on Ms. Pacman (C-D) performance with millions of training steps

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:

> The game state
> The search probabilities
> The winner
> Where each will be processed using the Monte-Carlo Tree Search (MCTS)



This is the plot for Muzero increase in rating with each training step in each game it played:

1) It is its graph in chess 2) this is its graph for shogi 3) this is its graph for Go 4) this is its graph for pacman.

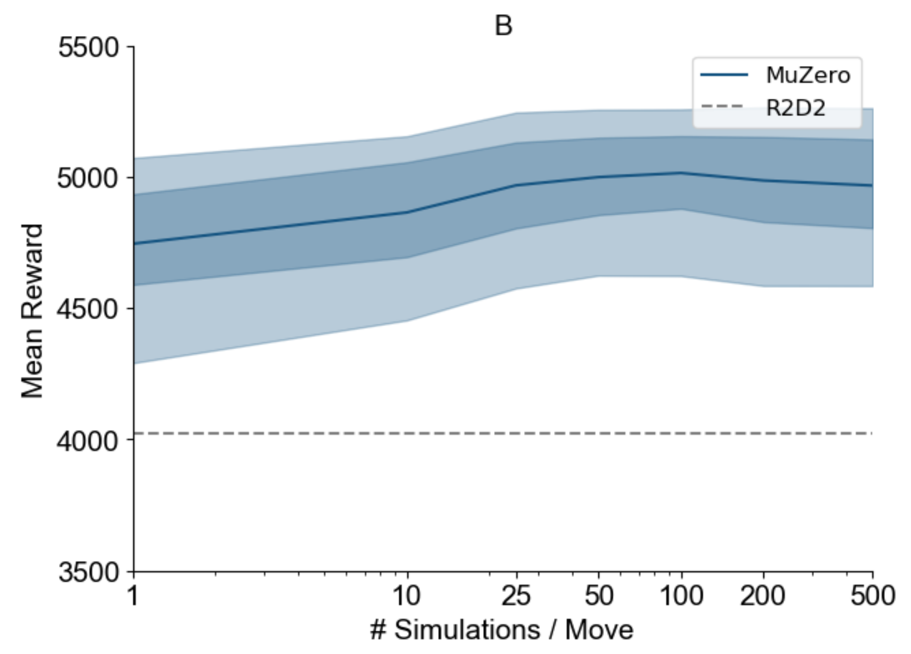
Data Modelling and Architecture

> 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

> in general, It has more blocks for more intensive games like StarCraft 2 and all the 57 Atari Development environment games

Performance on all 57 Atari games



Simulation

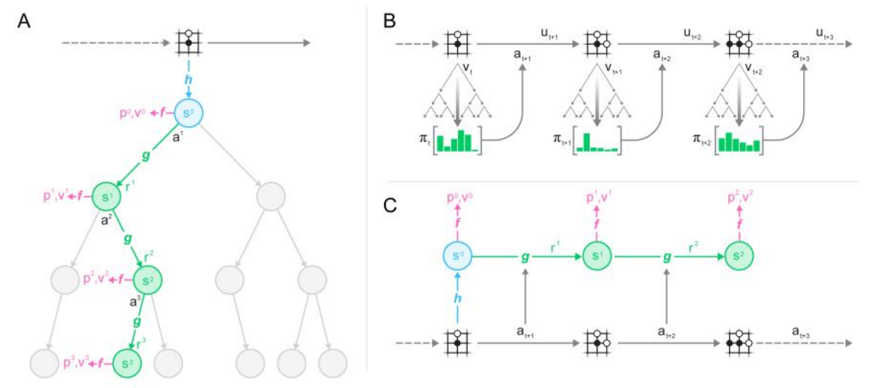
> It learns with 'tabula rasa' (i.e. with a blank slate). Unlike Alpha Go Zero, it is not even fed the rules of the game. Therefore, it has to figure out the rules as well as the condition which is considered a win, as it plays the game with itself, millions of times

> 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.

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

MCTS process



Project Plan

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

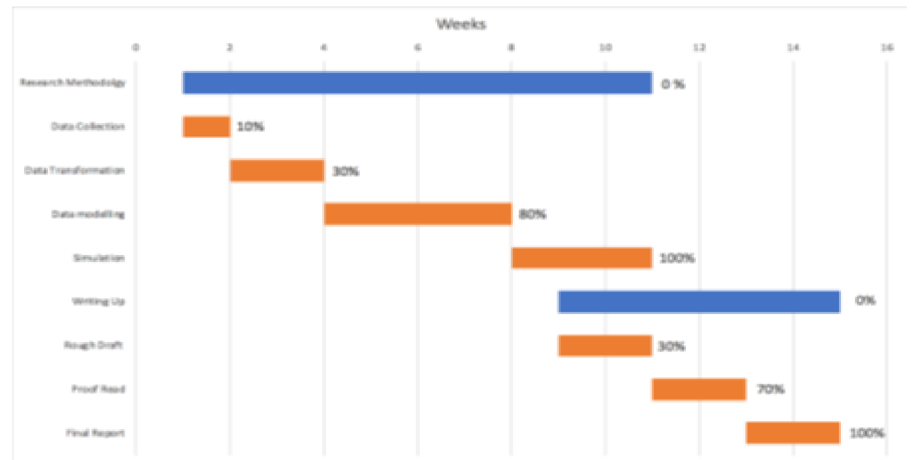


Photo caption here Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis vel massa eu ipsum tincidunt congue sit amet ac orci. Donec