Mastering Atari, Go, chess and shogi by planning with a learned model
dhr8

1. Introduction

AlphaGo, developed by Deepmind, was the first AI (Artificial Intelligence) programme to use neural networks and tree search to defeat humans in the game of Go [1]. Estimating a policy (mapping from state to action) and a value estimate (probability of winning from a given state) was used to accomplish this. AlphaGo utilized a lot of human knowledge, had many Go heuristics built into the agent and was well-versed with the game's rules [2].

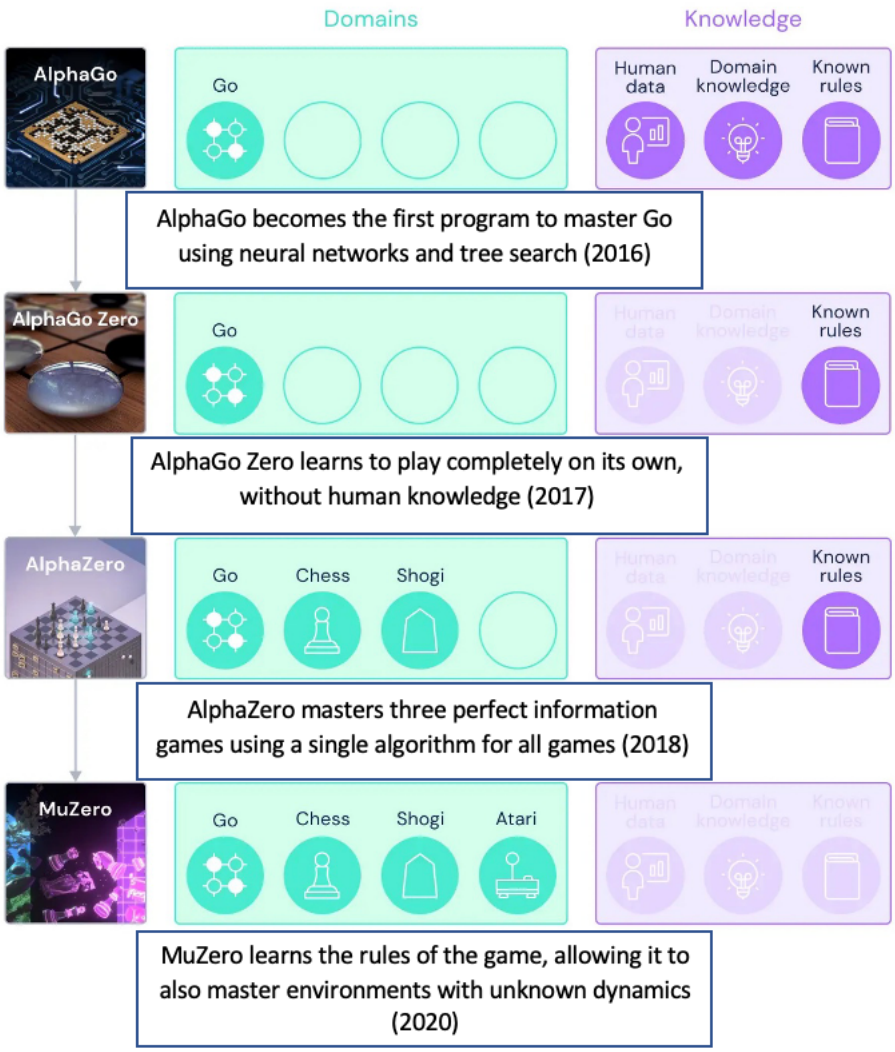

Figure 1: History of Deepmind Developments [1].

AlphaGo Zero was another model that learned to play the game without any prior knowledge of the game or human data, relying solely on self-play reinforcement learning. AlphaZero was the third model, which was a conceptual upgrade from AlphaGo Zero and could play Go, Chess, and Shogi. There was a need for a method that could learn a model that described their surroundings and then apply the model to choose the optimal course of action. MuZero recognises this problem by focusing on the most important components of the environment when planning. MuZero was able to set a reasonably high Atari benchmark while also matching AlphaZero's performance in classical planning challenges such as Go, chess, and shogi. It was used in situations where the game rules were unknown [1].

MuZero created its own dynamic model of the environment to learn the consequences of its decisions and forecast future actions without knowing the rule of the game [3]. MuZero's ability is unique to the model and the only important criteria is the agent's decision-making process and not the complete environment. It uses a deep learning framework to model three essential elements:

- Obtains a value that determines the accuracy of the current position from the ideal position.
- Obtains a policy which can schedule the optimal course of action in the present position.
- Obtains a reward from the previous action and analyzes how effective the action was.

Rather than collecting data from each environment, MuZero utilises this learning model to enhance its planning over time [4].

2. Main Solutions

MuZero is a model-based reinforcement learning algorithm, where it predicts the succeeding state and rewards it through its learned environment model. It learns a detailed predictive model that is used to simulate hypothetical scenarios. Reinforcement learning is a type of agent-based learning that learns by interacting with the environment. They are reinforced by rewards, which might be either scarce episodic rewards or intermediate rewards after n actions. We utilise the numbers 1,0,-1 to indicate positive, neutral, and negative feedback for actions in this model. MuZero also uses a modified version of the Markov Decision Process (MDP) model, which duplicates the underlying structure of MDP but lacks any environmental semantics. The MDP makes decisions in a discrete, stochastic, or sequential environment, however MuZero's modified MDP is deterministic instead of stochastic [5]. MuZero model consists of three main components:

- **A representation function(h)** which builds an initial internal state from the previous observations. The function takes the present and previous observations and produces a state which acts as an input to the other network. The model is trained to provide information for s_k to predict the future observations correctly [5].

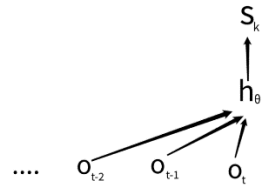

Figure 2: Representation function (h) [5].

- **A dynamic function(g)** is used to forecast the future internal state, act in the current internal state and immediately reward the action. The function uses $s_k$to return the next hypothetical state s_{k+1} and rewards estimated for the action r_{k+1}. The dynamic function must be executed for each action to predict the desired effect. The representation s_k should have the knowledge about the environment to assist us to predict the future rewards. Using the current state s_k, we can observe the reward in an instance and then choose various actions. We expect to obtain several rewards, which makes managing search space complicated. To manage the rewards, we utilise a prediction function [5].

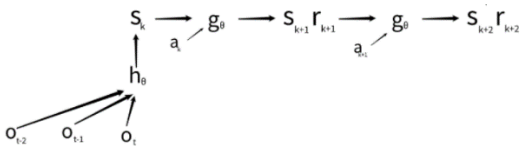

Figure 3: Dynamic function (g) [5].

- **A prediction function(f)** is used to create a policy and a value estimation from the current state. This state takes the state $s_k$and returns the result of value function $V_\pi(s_k)$ as well as distribution over actions for the policy $\pi(s_k)$. When the prediction function is called it arises our current situation as shown below:

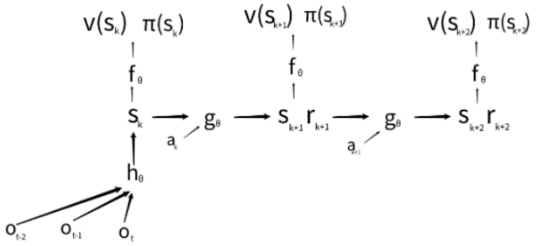

Figure 4: Prediction function (f) [5].

In the prediction function, we execute this particular part of the neural network whose $V_\pi(s_k)$ and actions are required which makes our planning easier. When the network has been properly trained, we can utilise the policy or state value function to take the appropriate action in the desired situation. The policy distribution immediately offers the most precise action, however the value function is more complicated to use. We can repeat the dynamic function for every possible value and use the state that the dynamic function emits to execute the state value function of that particular state. Furthermore, this allows us to decide the action which provides the largest state value associated with it [5].

The reward, the value function and the distribution over action are the network's output. Based on the prior states and actions, we can then assign the reward to take the correct value that it would take in that particular environment. We'd like the value function and distribution function to correlate a value with the optimal policy, which is determined on the same thing. Ideally, the value function and distribution function should correlate a value with the optimal policy, which is determined on the same model. This model is used to simulate predicted routes for use in Monte Carlo tree search [5].

Monte Carlo Tree Search (MCTS):

The Monte Carlo Tree Search is defined as a heuristic search algorithm for decision process. It is used to estimate the policy's course of action in order to increase and maximise the reward. The values reported by a badly trained neural network are not consistent. In MuZero, a Monte Carlo tree search produces:

- "State Value Estimates ($V(s_k)$) which are more consistent with later-in-time-state-value estimates, policies and rewards."
- "Policy Estimates ($\pi(s_k)$) which are more consistent with later-in-time-state-value estimates, policies and rewards."
- In Monte Carlo Tree Search, the predicted reward does not produce a new target [5].

Monte Carlo Tree Search only uses neural network to determine the goodness of state. Because the search produces more consistent policy and value functions, it can choose to utilise the new policy distribution to determine the appropriate action in a real environment. It might also employ the more stable policy distribution or value functions as a target for the neural network, allowing it to be trained better and efficiently. While training the MuZero model over time we observe:

- All the actions that the network does is random. But the neural network is designed to predict the reward r_k from the random actions which it encounters in the environment. The state value estimates ($V(s_k)$) begins to move towards a desired value for a random policy by the reward.
- The Monte Carlo Tree Search returns an improved policy (which is improved by learned values for r_t and $V(s_k)$) which is used in action selection. Which implies that the action selection is

better than the random selection and it is also used as a target to train the policy distribution, for it to move towards an improved random policy.

- That makes the target for $V(s_k)$ to change due to the state value function being policy specific. The higher accurate value for $V(s_k)$ is directly related to an improved policy. The state value function no more predicts the value for a random policy but for an improved random policy.
- The loop is continuous where the better estimates for the state-value functions and rewards propagate through the MCTS into a policy. These improved policies help MCTS to improve the depth of its search.
- Since the targets for $V(s_k)$ and π are generated using Monte Carlo Tree Search, makes the older historical data relevant since the network essentially predicts the right value if it had prior experience with the current state. The process eventually converges on the optimal policy over a longer duration [5].

Working of MuZero model:

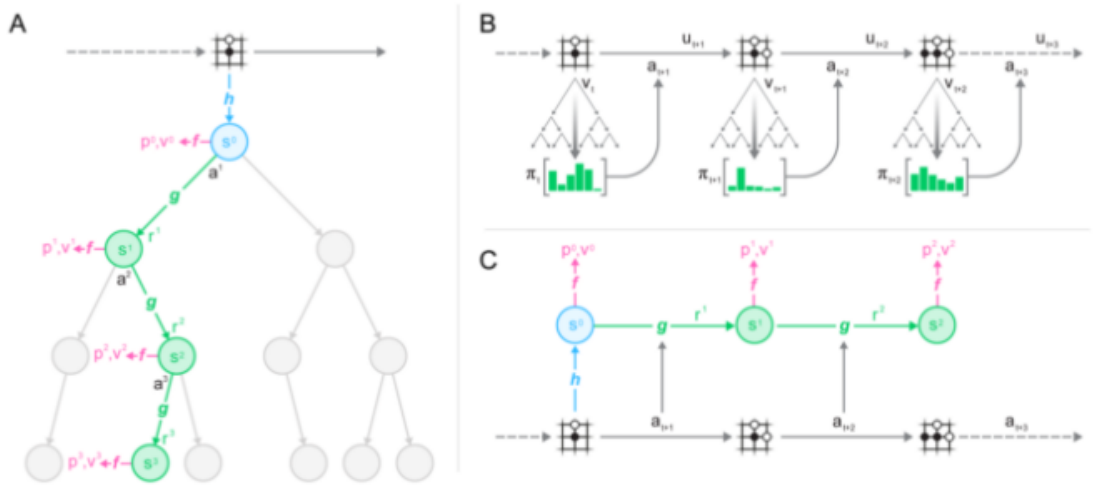

Figure 5: Planning, acting and training with a learned model [6].

Diagram A represents how MuZero uses its model to plan. The sequence of using a representation function (h) to map raw observation into a hidden state represents tree-based planning [2]. In MuZero, the policy network is represented in this space (hidden state), so instead of matching the raw observation with the actions or value estimates, it rather takes these hidden states as inputs. The hidden state computes the policy and value function by prediction function f. Whereas the dynamic function (g) learns to map the action and the hidden state to an estimated hidden state [6].

Diagram B depicts how MuZero acts in the environment. The policy network is trained similarly by copying the action distribution which is produced by Monte Carlo Tree Search. MCTS is carried out at every timestep 't' [2]. The search policy samples an action, which is related to every action of visited count from the root node. The action is received by the environment and new observations and rewards are generated. The replay buffer stores the trajectory data at the end of the episode [6].

Diagram C represents the training of the system and how is it done. Each of the three neural networks is trained using a joint optimization of the difference between the actual return with the value network. The replay buffer samples a trajectory [2]. For the initial step, inputs (from past observations) from the selected trajectory are received by the representation function (h) This is followed by

unrolling the K steps recurrently. For every step k, the dynamics function (g) receives an input from the hidden state from the last step and real action. The parameters from all the three functions are trained together, end to end by backpropagation with time, to predict the policy, value function and reward. At the top of the figure, the sequence of observation of the schematic Go boards is represented [6].

3. Technical and Ethical Issues

In general Machine Learning technique used in this model, has many technical challenges which occur due to:

- **Poor quality of data:** Which can cause inaccurate results and makes the whole process exhausting.
- **Overfitting of Training Data:** This occurs when the model trains with an enormous amount of data that affects its performance negatively. We can tackle this issue by analyzing the data with more complexity, removing any outliers during training, using a model with fewer features and using techniques such as data augmentation. Data augmentation is a method which significantly increases the amount of data by adding copies of a slight modification of existing data. [7]
- **Self-Supervised Consistency Loss:** MuZero has a problem like any other model-free learning method, it does not start to learn until it receives non-uniform reward values.
- **Value Prefix:** Many Reinforcement Learning based algorithms try to predict the future at the exact moment, which is very complex. MuZero model tries to predict the reward at a particular time step into the future. Which causes wastage of computational power since the knowledge of all the rewards is not required. Even the uncertainty about a particular frame where pleasure and pain occur. It affects the used search with the Monte Carlo Tree Search, which depends strongly on the future predictions of MuZero's model of when the pleasure or pain occurs [8].

MuZero's has a general AI agent. It learns by itself to outperform humans at Atari games and it uses Reinforcement learning to achieve it, which is real-time observation-based. The model is based on learning through trial-and-error interactions with its environment. The agent is used to train to implement a sequence of the decision on the action within its environment achieving maximum reward to win the game. The application of such algorithms in robotics and industrial control will expand automation and cause a high risk of unemployment. Reinforcement learning is a great solution for open-ended, real-world problem solving which makes it an AGI (Artificial General Intelligence) that raises concern among researchers about the growth of truly autonomous AI which does not go together with human values. Autonomous agent learns to play games in the virtual world by itself and an intelligent but rouge AI putting humanity at existential risk lies a multitude of sociotechnical concerns [9].

The use of RL in medical treatment protocols is increasing in popularity and it comprises of a sequence of decisions (which treatment options to try) with uncertain outcomes (it may vary with different types of people) and it is all connected eventually to the patient's health. The patient's data such as medical history, clinical charts and doctor's notes are not considered as MuZero does not use data and tries to learn by itself. The goal of the system was to predict treatment plans which can lead to severe consequences and can even cause deaths. The medical industry is developed with this fundamental uncertainty. With strict codes of ethics, clinical trials and industrial standards, the field is developing to reduce harm. The Reinforcement Learning model must learn to develop within this infrastructure. This can also be applied to social media, but the predicted harm is very hard to consider and

theoretically, it has no regulatory scaffoldings. It was also applied to YouTube search and recommended algorithm which caused a rabbit hole effect i.e., trying to maximise watch time which made the users addicted [9].

Alpha C2 has a similar architecture to MuZero which is used as an intelligent air defence commander. It was developed to fight against enemy aircraft and intercept missiles. However, wars tend to cause human causalities. When automation is in control of action, it can potentially claim civilian lives, who will bear the responsibility?

4. Alternative Approach

Reinforcement Learning agents can decide the future predicted moves by using a model of the environment for board games like Go, Shogi and Chess while keeping the sample efficiency relatively high. In MuZero, the learning in the environment happens dynamically, assigning the agent to various tasks and achieving state-of-the-art performance. MuZero uses a state representation that is internal and obtained from the real environment for predictions. In this approach, there is a combination of the model's internal predicted state representation to the state in the environment using extra terms such as a reconstruction model loss and an easier consistency loss which works unsupervised and independently while acting as a constraint to normalize the process of learning. This method achieves a new integration of reconstruction model loss and an easier consistency loss yields a big performance upgrade in OpenAI Gym environments. The modification also allows self-supervised learning pertaining to MuZero, which can make the algorithm learn about the dynamics of the environment before setting a goal [10].

Two changes are proposed to the MuZero algorithm which may increase the performance and these proposals are based on the ideology that MuZero is very unconstrained in its embedded state representation which makes the algorithm flexible but can cause significant harm to the process of learning. The changes are done by individually weighable loss terms so that it can be viewed as a generalized MuZero algorithm and can be used together or independently.

- In the first proposed change, a reconstruction function is introduced (h_θ^{-1}). The function performs the inverse operation on the representation function by mapping the internal state to real observations instead of mapping observations to the internal state (in the representation function) which makes the function perform a generative task. The function used is an approximation that can affect the reconstructed observations to reduce their accuracy. Since this embedded state has very less information than the real observations. It is virtually impossible for the reconstruction function to get back what has been deleted by the representation function. It ultimately means the observation is an estimation of real observations and the function is deleted once the training process is over, but it affects the learning for the representation function and dynamics function. The advantage of this change is that it makes the ability to pertain an agent self-supervised [10].
- In the second proposed change, a simple loss term is introduced for MuZero's loss equation i.e., consistency loss and does not need an addition function to be in the system. This occurs due to possible inconsistencies during the embedded state representations after every action of the dynamics function. While running the model through the consistency loss function it is forced to maintain relevant information to predict the next state even in surrounding where the rewards are randomised [10].

5. Limitation and Proposed Direction

MuZero achieved superhuman performance in planning algorithms for the board games (Go, Chess and Shogi) and had outperformed state-of-the -art model-free Reinforcement Learning algorithms. MuZero is great at working in a large, combinatorial environment but to solve many real-world applications it is required to work in multiple environments which it is not designed for because the games tend to work in a single known environment [6]. Since the algorithm uses Reinforcement Learning the sample efficiency remains a key challenge. MuZero does not work in a stochastic environment [8].

Deepmind proposes to use MuZero for various applications beyond video compression and in a variety of research environments for Reinforcement Learning agents to resolve various real-world problems. They intend to propose a single algorithm that can learn and make a variety of encoding decisions for optimal rate-distortion trade-off. Their goal is to build a single algorithm to optimise thousands of real-world systems and make the computational process faster, automated and less intensive [4].

6. References

[1] Julian Schrittwieser, et.al., "MuZero: Mastering Go, chess, shogi and Atari without rules," Deepmind, 23 December 2020. [Online]. Available: https://www.deepmind.com/blog/muzero-mastering-go-chess-shogi-and-atari-without-rules.

[2] C. Shorten, "The Evolution of AlphaGo to MuZero," Medium, 17 January 2020. [Online]. Available: https://towardsdatascience.com/the-evolution-of-alphago-to-muzero-c2c37306bf9.

[3] D. Foster, "MuZero: The Walkthrough (Part 1/3)," medium, 2 December 2019. [Online]. Available: https://medium.com/applied-data-science/how-to-build-your-own-muzero-in-python-f77d5718061a.

[4] A. Gopani, "Is DeepMind's MuZero ready for the real world?," Analytics India Magazine, 24 February 2022. [Online]. Available: https://analyticsindiamag.com/is-deepminds-muzero-ready-for-the-real-world/.

[5] 1a3orn, "EfficientZero: How It Works," Alignmentforum.org, 26 November 2021. [Online]. Available: https://www.alignmentforum.org/posts/mRwJce3npmzbKfxws/efficientzero-how-it-works#3_2__Monte_Carlo_Tree_Search.

[6] J. Schrittwieser, et.al., "Mastering atari, go, chess and shogi by planning with a learned model," *Nature,* vol. 588, pp. 604--609, 2020.

[7] Vanshika, "7 Major Challenges Faced By Machine Learning Professionals," geeksforgeeks, 13 October 2021. [Online]. Available: https://www.geeksforgeeks.org/7-major-challenges-faced-by-machine-learning-professionals/.

[8] Weirui Ye, et.al., "Mastering atari games with limited data," *Advances in Neural Information Processing Systems,* vol. 34, 2021.

[9] T. Zick, "Between Games and Apocalyptic Robots: Considering Near-Term Social Risks of Reinforcement Learning," medium, 16 April 2021. [Online]. Available: https://medium.com/berkman-klein-center/between-games-and-apocalyptic-robots-considering-near-term-societal-risks-of-reinforcement-35ec5af8f04a.

[10] Julien Scholz, Cornelius Weber, Muhammad Burhan Hafez and Stefan Wermter, "Improving Model-Based Reinforcement Learning with Internal State Representations through Self-Supervision," in *2021 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2021, pp. 1--8.