MuZero: Overview And Improvements for Dynamic Gaming Systems using Error Improvements via Self-Supervision

Aishwarya Singh
Katz School of Science and Health
Yeshiva University
New York, NY, United States
asingh10@mail.yu.edu

Atreish Ramlakhan
Katz School of Science and Health
Yeshiva University
New York, NY, United States
ramlakha@mail.yu.edu

***Abstract*—There was a pivotal moment in artificial intelligence history when the AlphaGo algorithm beat the world master Fan Hui in the board game Go in a five-game match. AlphaZero was developed in response, by the AI research company DeepMind to master chess, shogi and go. AlphaZero has hard coded rules for setting search hyperparameters and the neural network is updated continually. MuZero is the offshoot of these algorithms and a game changer in that it does not require the input of the rules of the game.**

***Index Terms*—reinforcement learning, artificial intelligence, AlphaGo, AlphaZero, MuZero, game-playing agents, Atari, Go, Shogi**

I. Introduction

MuZero is a model-based reinforcement learning algorithm that first learns a model's environment, thus, it is a learned model. It uses tree-based search to determine the best action, rewards, and trajectory. In doing this iteratively over and over, it learns the model and then plans with respect to the learned model. Model-free reinforcement learning is most successful, it estimates the optimal policy and/or value function directly from interactions with the environment instead of having to have the model as an input. However, this approach is subpar for domains that require precise and sophisticated lookahead strategies such as Chess and Go. Now, this is where MuZero is different, it is a model-based reinforcement learning technique and builds upon the search and search-based policy iteration algorithms and incorporates the learned model to the training procedure.

The main idea behind MuZero is to predict those aspects of the future that are directly relevant to planning. The model receives the observation e.g. image of a Go board or Atari screen as an input and transforms it into a hidden state. The hidden state is then updated iteratively by a recurrent process that receives the previous hidden state and a hypothetical next action. At every step, the model predicts policy, value function, and the immediate reward. The model is trained end to end with the sole objective of accurately estimating these 3 important qualities. There are no requirements for the hidden state to capture all information necessary to reconstruct the original observation. Also, there is no requirement that the hidden state matches the true state of the environment. These provide a very open environment for the model to learn.

MuZero is quite a different approach to model-based reinforcement learning that is focused on predicting the value function in an abstract Markov Decision Process. The main idea is that the value generated in the abstract mimics the value the model would obtain in the real environment. The MDP is viewed as a hidden layer in a deep neural network.

II. Literature Survey

In the 2020 paper **"Mastering Atari, go, chess and shogi by planning with a learned model"** the authors introduce the MuZero algorithm and its innovation to the central task of reinforcement learning methods applied to games. By combining a tree-based search with a learned model, the MuZero algorithm achieves superhuman performance in a range of challenging and visually complex domains. There is no requirement that the algorithm has any knowledge of the environment's underlying dynamics. MuZero uses internal state representations derived from real environment states for its predictions. The MuZero algorithm learns an iterable model that produces predictions relevant to planning: the action-selection policy, the value function and the reward. Upon its application to Atari games, where many artificial intelligence models are tested. MuZero performs incredibly well.

In the same year the 2020 paper **"The Value Equivalence Principle for Model-Based Reinforcement Learning"** describes the theoretical and practical benefit of a major principle used in the 2021 paper concerning the value equivalence principle. Two models are value equivalent with respect to a set of functions and policies if they yield the same Bellman updates. The paper proposes a formulation of the model learning problem based on the value equivalence principle and analyzes how the set of feasible solutions is impacted by the choice of policies and functions. The set of policies and functions considered is augmented and the class of value equivalent models shrinks, leading to a collapse to a single point corresponding to a model that perfectly describes

the environment. In many problems, directly modeling state-to-state transitions may be both difficult and unnecessary. By leveraging the value-equivalence principle one may find simpler models without compromising performance, saving computation and memory.

In 2021, there was a coordinated effort by researchers to further improve on the MuZero algorithm by adjusting the error or L2 Norm used in the planning process. The paper entitled **"Improving Model-Based Reinforcement Learning with Internal State Representations through Self-Supervision"** details this process by binding the model's predicted internal state representation to the environment state via two additional terms: a reconstruction model loss and a simpler consistency loss, both of which work independently and unsupervised, acting as constraints to stabilize the learning process. This new integration of reconstruction model loss and simpler consistency loss provide a significant performance increase in OpenAI Gym environments.

In a final note, another 2020 paper is of use in contextualizing MuZero in the wider world of model-based learning methods. **"The LoCA Regret: A Consistent Metric to Evaluate Model-Based Behavior in Reinforcement Learning"** details a way to compare different reinforcement learning algorithms, and particularly model-based approaches such as MuZero, in the domain of sample efficiency and adaptability to local changes in the environment. The paper primarily defines a metric to quantify such concepts as the 'Local Change Adaptation regret'. Importantly, it also explains a test of MuZero on a Mountain Car based experiment against a control scenario using a SARSA(λ) method. This experiment shows that while MuZero substantially outperforms the simpler model-free method, it is still not achieving ideal model-based performance and there is still room to improve.

III. Motivation

Within the field of Artificial Intelligence, gaming and the ability of algorithms to learn and compete with masters has long been sought out as a milestone. The ability for artificially created algorithms has both fascinated and terrified humans since the science fiction classics of the 1950s. MuZero exists as the continuation of algorithms that have played and won against human masters in games such as Chess, Go and Atari. In 2015 AlphaGo became the first program to beat a human master player at a full game of Go on a 19x19 board. In late 2017 AlphaZero was introduced to the world as a successor to AlphaGo that generalizes the learning process to Go, Chess and Shogi. MuZero was introduced in 2019 that succeeds AlphaZero, it improves on most games, improves on Go performance, and is able to play complex visual based games such as all 57 Atari games, it uses 20 percent less computation and enhances performance. We hope that the research identified can contribute to the algorithm's continued success and optimality. We will demonstrate how MuZero approaches a problem, and how new research into varying methods of error calculation can and do further improve performance of the algorithm when used in Atari games.

IV. Proposed Methodology

The dynamics of the MuZero algorithm can be described with the following steps that visualize the planning, acting, and training of a learned model.

The initial step shows how MuZero plans with Tree Search and simulation of future trajectories. We have our observation, the black dot, and it is fed into our representation function h. The is a deep neural network that maps from the observation space to the hidden space. We are given a hidden state; s_0. This state is fed into a predictive function f, that outputs the v_0; the value function at that state and its rewards average, and p_0 the policy predictions i.e., how to act at that state. These two outcomes determine a_1 which is an action that produces another state by applying g and so forth. Using tree search gives us an approximation of the immediate next states and actions and captures their values v_1, v_2, v_3... Throughout the tree, we simply choose actions a_i that leads to the highest values at v_i.

At the end of the tree, we have a histogram that provides insight into the values. Here, the model learns what action at time t to take that maximizes the reward u_{t+1}. Then, the observation we have after a_{t+1} is fed into the process again and the model repeats. At the end of the episode the trajectory data is stored into a replay buffer.

In the final phase, we train our networks such that given a trajectory and action sequence we know what individual g- function neural network, v_1-value function, p_1-approximate policy prediction to run the tree search to match as close as possible. What we want is the p_0 to match as closely to a_0, a very accurate policy.

Our main objective in this experiment is to improve the MuZero algorithm further with newly developed theoretical improvements and practical research improvements which are detailed in the paper "Improving Model-Based Reinforcement Learning with Internal State Representations through Self-Supervision".

These new methods of calculating error loss by binding the model's predicted internal state representation to the environment state via two additional terms: a reconstruction model loss and a simpler consistency loss, significantly improves on MuZero's performance when learning the Atari games. Both improvements work independently and unsupervised and also act as constraints to stabilize the learning process. The reconstruction function introduced performs the inverse function of the representation function. It maps internal states to observations, thereby performing a generative task. This is then optimized as an auxiliary task, but this is likely to not be perfect since we are using a learned dynamic model.This reconstruction function is also trained with an additional term in the MuZero loss equation. Since we are comparing real observations to the reconstructed observations, the better the model (accuracy) the lower the loss term.

V. Experimental Design

We train a MuZero agent based in PyTorch as prescribed by Schrittwieser et. al. on the common reinforcement learning

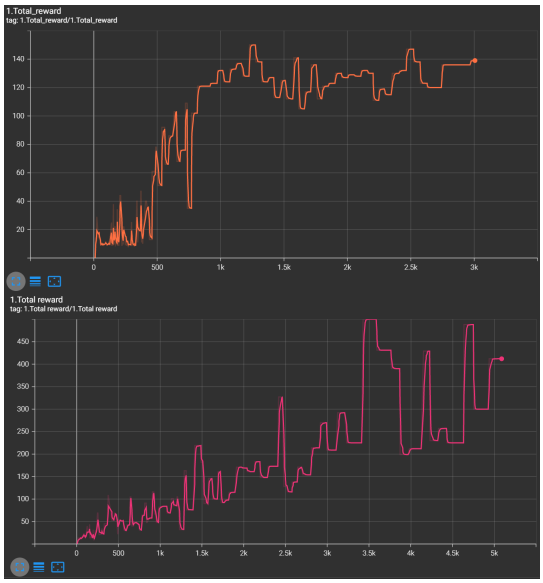
'gym' problem called Cartpole, wherein the agent tries to balance an unstable inverted pendulum for as long as possible. Cartpole is chosen over Atari games due to ease of implementation and speed of useful training for comparison.

Using the default MuZero agent as a control we also train a variant with the reconstruction loss term introduced by Scholz et. al. on the same Cartpole problem with the same domain parameters. This gives us a comparative study that we shall elucidate below.

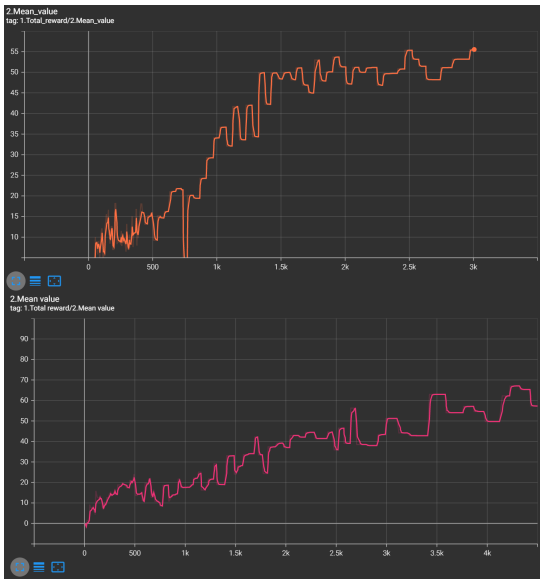
VI. Results and Comparative Study

The upper orange plots represent MuZero in its original form, while the lower pink plots represent MuZero augmented with a reconstruction loss term. The x-axis in all plots represents total episodes, with the augmented version being run for a longer total duration than the baseline. All results are extracted from Tensorboard displays.

A. Total Reward



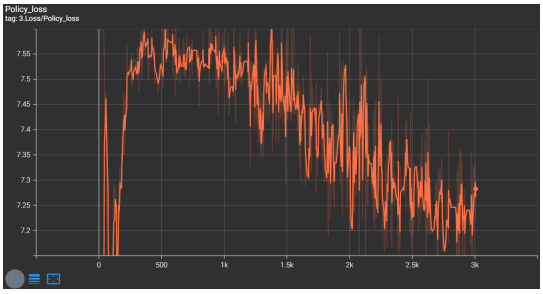
B. Mean Reward

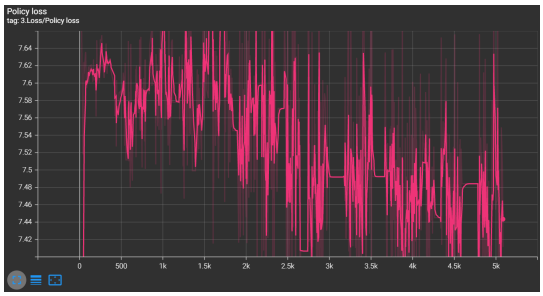


C. Episode Length

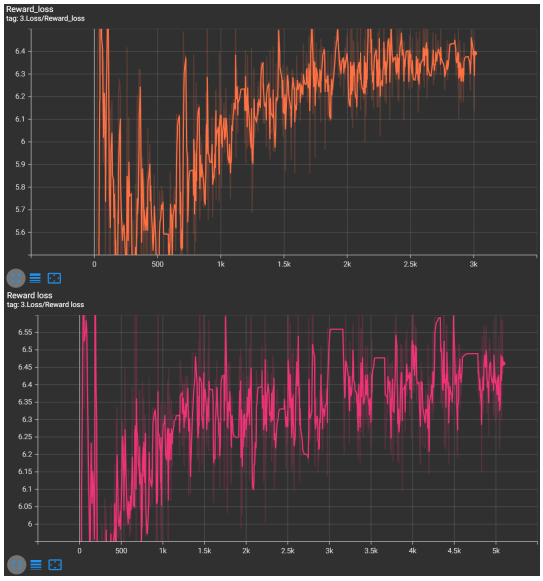
D. Total Weighted Loss

E. Policy Loss





F. Reward Loss



G. Value Loss

H. Commentary

We observe that the augmented version of MuZero is able to achieve higher total rewards in a similar number of episodes than the original version. In particular, the augmented MuZero is able to extend episodes for much longer than the original, which in the case of the Cartpole case is a very positive sign.

VII. Future Work

MuZero demonstrates the continuing evolution of the quest for Artificial Intelligence supremacy in the realm of gaming. The original AlphaGo algorithm was revolutionary in and of itself because of its ability to learn from the dynamics of the game. In the game of Go, we have more possible options to play than there are atoms in the known universe. Thus, no computer can learn all moves. The ability for an artificially crafted algorithm to win at Go and other board games is surreal. MuZero has now expanded the scope of where the algorithm can learn with board games and semi-graphic Atari games.

This was further improved with error improvements in learning with new literature using self-supervision. We have shown here that MuZero can be improved upon with the addition of loss terms in end-to-end training that help keep its value-equivalent internal representation on track. One area we were not able to test however was the Local Change Adaptation (LoCA) regret for the version of MuZero that was enhanced with the reconstruction loss term.

Seijen et. al. showed that while the default MuZero was very good in terms of sample efficiency and model-based behavior in changing environments, MuZero was still not able to attain theoretical ideals in terms of their LoCA regret metric. It is possible that the improved variant would have made a step closer to ideal in this regard, but experiments testing that possibility are a subject of future work.

Other areas of improvement we suggest are more practical in that, for students and researchers who aim to improve on this topic, the repositories should become more accessible. There is a lack of sufficient documentation in the official MuZero repositories, insufficient version online, cumbersome package requirements and the non-use of Ray in our particular case. Ray is a package used for supercomputers, but in our local environment we have no need for such computing power and thus, the debugging necessary is quite time consuming and confusing.

References

[1] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver, “Mastering Atari, go, chess and shogi by planning with a learned model”, Nature, 2020, vol. 588, pp. 604–609.

[2] J. Scholz, C. Weber, M. B. Hafez and S. Wermter, ”Improving Model-Based Reinforcement Learning with Internal State Representations through Self-Supervision”, 2021 International Joint Conference on Neural Networks (IJCNN), 2021, pp. 1-8.

[3] H. Van Seijen, H. Nekoei, E. Racah, S. Chandar, “The LoCA Regret: A Consistent Metric to Evaluate Model-Based Behavior in Reinforcement Learning”, Advances in Neural Information Processing Systems, 2020, vol. 33, pp. 6562–6572.

[4] C. Grimm, A. Barreto, S. Singh, D. Silver, “The Value Equivalence Principle for Model-Based Reinforcement Learning”, Advances in Neural Information Processing Systems, 2020, vol. 33, pp. 5541–5552.

[5] https://www.youtube.com/watch?v=We20YSAJZSE

[6] https://www.youtube.com/watch?v=L0A86LmH7Yw

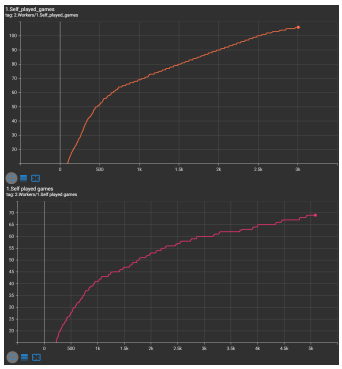
[7] https://github.com/geohot/ai-notebooks/blob/master/muzero_tictactoe.ipynb

[8] https://www.youtube.com/watch?v=lVMgxtm5L-U

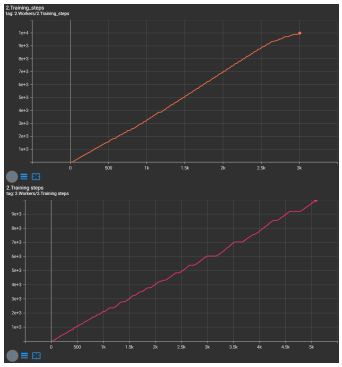
[9] https://www.youtube.com/watch?v=yIrFIOx4VP8&t=1197s

VIII. Appendix

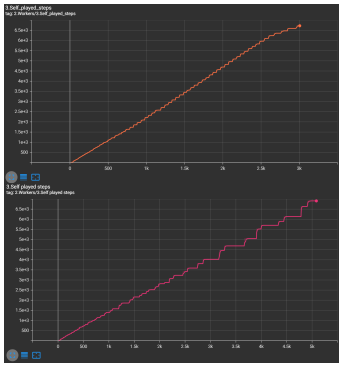
A. Additonal Figures: Self-Played Games



B. Additonal Figures: Training Steps



C. Additonal Figures: Self-Played Steps



D. Additonal Figures: Learning Rate