**Reinforcement Learning in Super Mario Bros.**

Kollabathula Preetham

**ABSTRACT**

The game Super Mario Bros is a very popular action game featuring a "real-life" scenario and a huge state space. It is one of the best game environments to learn about the different reinforcement learning algorithms. The AI bot must deal with many items and challenges in the Super Mario Bros environment, supporting a knowledge-rich approach to the learning process.

The goal of this project is to use Deep Reinforcement Learning technique called Proximal Policy Optimization to train an agent to self-play Super Mario Bros. Using Proximal Policy Optimization, our AI agent will learn how to play by watching only the raw game pixels, we utilize convolutional layers early in the network, followed by dense layers to retrieve our policy and state-value output. We use Stable-baselines3 library to import the PPO reinforcement learning algorithm we. The aim of this project in layman terms is to, start on the left edge of a level and travel to the flagpole without dying while maximizing the score. The purpose of this project is to construct an AI bot that can effectively explore a single level of Super Mario.

***Keywords:*** Reinforcement Learning, Stable-baselines3, OpenAi Gym, PPO, Convolution Neural Networks.

1. **Introduction**

So, in the game Super Mario, we play as Mario, and the objective is to race through the Mushroom Kingdom by evading or eliminating the enemies and save Princess Peach. Mario can jump on his enemies to eliminate them. On his journey to princess peach, we can collect various power-ups. Some of these power-ups are:

* The flower, transforms Mario into a Fiery Mario, allowing him to throw fireballs.
* The super star, temporarily transforms Mario into an Invincible Mario, granting him invincibility and allowing him to defeat most enemies by touching them.
* Hitting the Question block brick will produce a mushroom which Transforms Mario into an Adult - Mario, allowing him to destroy Brick Blocks.

A picture containing text, toy, sign

Description automatically generated   A picture containing text, sign

Description automatically generated

Regular Mario Fiery Mario Invincibility Star Adult Mario

Mario dies if he touches an enemy. If we are in the Adult or Fiery Mario state, and then a enemy attacks Mario, then we transform back to the regular Mario and the game proceeds normally. But, if the enemy attacks Mario in his regular form, or if we fall down a pit or if the clock timer runs out then we lose one of the three lives and then the game restarts from the beginning or the checkpoint. Mario must typically move right to succeed; yet, just moving right is far from ideal policy. Mario must dodge and/or destroy opponents, leap over cliffs and pipes, gather mushrooms and coins, and stay within the time restriction to perform optimally. These obstacles greatly complicate the work at hand.

## A. Reinforcement Learning

One of the most interesting topics in machine learning is reinforcement learning. The main idea is that an agent tries to learn, by interacting with an uncertain environment and using the experience gathered, to optimize some goal given in the form of cumulative rewards. Reinforcement learning algorithm is one type of machine learning algorithm in which the state of the model is fed to the learning agent which interacts with the environment by performing the best action for the given state and then based on the output gives out a reward or penalty to the agent. By doing this, it enables the RL agent to automatically select the best behavior in each circumstance to enhance the performance by maximizing the reward function. It is a machine learning training strategy that rewards positive actions while penalizing undesirable ones. 

Diagram

Description automatically generated

A reinforcement learning agent, in general, can detect and comprehend its surroundings, act, and learn via trial and error. Ai can do some cool things when it comes to playing games. The awesome thing about reinforcement learning is that it can be used to play games in a whole heap of different open world environments now one of the games that it just so happens to be pretty good at playing is Super Mario. So, the objective of this project is to create an AI bot which can learn to complete the game on its own.

## B. Proximal Policy Optimization

## The impetus to use PPO to solve many issues stems from PPO's recent spike in popularity and efficacy in obtaining cutting-edge performance in several RL benchmarks (Schulman et al., 2017). PPO is a policy gradient technique family that performs each policy update using multiple stochastic gradient ascents. The goal of PPO is to build a policy gradient estimator and input it into a stochastic gradient ascent. To avoid the huge policy updates that occurred in the policy gradient, it proposes a clipping function to improve training stability by restricting policy change at each step. PPO computes the ratio of the present policy to the previous policy:

A picture containing text

Description automatically generated

## C. Open AI Gym Environment

## Super Mario Bros is one of the games provided through OpenAI Gym. Super Mario, is a fascinating domain for RL since it has a vast state space yet simple dynamics. The game’s numerous items are great for motivating the RL agent to learn by interacting with them in order to attain its goal. Super Mario is an appropriate platform for training an RL agent since it can be expressed as a Markov Decision Process.

## Chart Description automatically generated

## • States: A succession of frames, each of which is a matrix built from the tiled interface of the OpenAI Gym emulation. At the start of each timestep, the Super Mario Bros. environment generates an observation. This observation is that a picture is always made up of pixels, and each pixel is represented numerically. The visuals on the screen employ RGB values and are 320 pixels wide and 224 pixels tall. In the implementation, we used the state to express observation.

## • Actions: It is a series of action combinations based on heuristics that span numerous frames. The Super Mario environment adheres to the set of actions available via the Nintendo Entertainment System console's controller. At each timestep, the agent performs an action based on the NES console's button combinations. The Super Mario environment turns binary encoded actions into discrete actions. As a result, the action space is as follows:

## Text Description automatically generated

## • Reward: Moving right will give the agent a positive reward, moving left will give the agent a negative reward, a death penalty with a negative reward for dying, and a negative reward if the agent takes longer to finish the level. During an episode, the environment will offer the agent a reward, and the reward will accrue until the done function returns true, indicating that the agent has either completed the level or utilized all three lives. The goal of the game is to go as far to the right as possible while avoiding death and doing it in a short amount of time.

**2. Related Work**

Liao and Yang (2012) employed an epsilon-greedy exploration strategy to apply the Q-Learning methodology to Super Mario. The author established four measures to evaluate performance: the total reward obtained by the agent given specific iterations, the likelihood that the agent would beat the level, the percentage of monsters slain, and the time spent on the level. The agent was able to train an agent that has a high killing rate and regularly beats the first level of World 1 by providing a reward mechanism for killing an adversary. The authors proved that by setting reward functions for doing specified actions, the agent may learn to perform additional tasks such as killing adversaries.

Grand and Loughlin (2018) compared the performance of many different Reinforcement Learning algorithms like, Q-Learning, Approximate Q and SARSA in the Super Mario Bros. environment. The Super Mario Bros. agent necessary to finish World 1's initial level. The heuristic agent's behavior was hardcoded to avoid gaps and enemies. The RL agents were unable to outperform the heuristic agent. The authors' most useful discovery, however, was the introduction of a performance matrix based on the mean and maximum distance reached by the agent given specific repetitions. This matrix was a decent predictor of the RL agent's performance.

A Q-Learning technique is one approach for implementing the RL algorithm. Q-Learning differs from a policy gradient technique in that, rather than employing a policy to map states to actions, Q-Learning learns a single deterministic action from a discrete collection of actions by finding the maximum value. Mnih et al. (2013) developed Deep Q Network, a convolutional neural network trained utilizing a Q-learning variation. An RL ATARI 2600 agent was trained, and it produced impressive results.

PPO was used by Beysolow II (2019: 38) to train an agent that can self-play the first level of World 1 Super Mario. The author adopted the generic PPO technique of Schulman et al. (2017) and preprocessed the photos by shrinking and converting the color to greyscale. The author did not specify any results because the implementation was for illustrative purposes; nonetheless, the author concluded that PPO training for Super Mario, will normally yield a positive outcome after 12 hours of training.

Most of Super Mario Bros. related studies were carried out in the first level of World 1. Even though Beysolow II (2019: 38) integrated PPO with Super Mario, there was no adequate investigation and analysis of the agent's performance in the first level of World 1 utilizing PPO. Nichol et al. (2018) explicitly outlined the method of teaching the agent to play Sonic the Hedgehog levels. However, no comparable activities or analyses were carried out at different levels of Super Mario.

**3. Data Preprocessing**

**A. Image Preprocessing and Frame Stacking:**

Our AI model uses photos of Mario to learn; the color image has three times as many pixels to analyze. Working directly with raw Super Mario bros. environment frames, which are 320x240 pixel pictures, is computationally intensive. Therefore, according to the parameters presented by Mnih et al. (2013). converting it to greyscale reduces the amount of input from which our model must learn. Frame stacking provides context for our AI, by stacking successive frames, we essentially provide memory to our AI model. To begin, we will import our frame stacking, gray scaling wrapper, and vectorization wrapper. I cropped the photos to 84 × 84 pixels and converted them to greyscale using the gym wrapper class package. The cropping was adequate to capture the action while still providing enough information for the PPO model to comprehend the frames. I also applied frame stacking to the environment since an agent activity is typically finished in four frames. The RL model gives the correct action for the agent by stacking the frames.

**B. Skip Stochastic Frame:**

I applied the stochastic frame skipping presented by Nichol et al. (2018) to inject a little degree of randomness accepted by the agent. By applying n actions over 4n frames, stochastic frame skipping brings unpredictability into the agent's activity. The probability of 0.4 was utilized to postpone the previous action by one frame and apply it to the present action.

Graphical user interface

Description automatically generated with low confidence Graphical user interface

Description automatically generated with low confidence

What we humans see How the AI model sees

**4. Approach:**

In the project, the first section is focused on the environment settings, which include specifics about the environment of Super Mario environment which is provided by OpenAI Gym and gym-super-mario-bros.

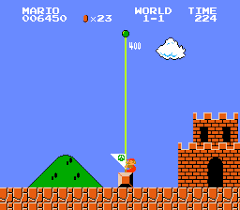
I have preprocessed the environment to remove extraneous details and enhance the environment. Data preprocessing is vital because the data supplied by the environment must be ideal for the agent training utilizing the PPO algorithm, hence it improves the agent's performance and shortens the training time.

The second part is focused on the implementation of the PPO algorithm and saving the trained model. I have used the stable-baselines3 library to import the PPO algorithm which is the core logic for the project. The data which is processed from the Super Mario environment is fed into two Convolutional Neural Network using the PPO CnnPolicy. The model is trained until the maximum number of episodes was reached.

I have trained the model for 9 million episodes which successfully crushed the first level of World 1. It took 4-5 days of training time. The time it took to complete training was recorded to test the algorithm's speed in training the agent. The shorter the time required, the better the algorithm performs. Later I plan on implementing the same project using different algorithms to compare the run times and reward functions using Tensorboard.

**5. Conclusion**

I have successfully trained an AI bot to play the first level of Super Mario Bros. using PPO algorithm. We also included data preprocessing to boost the agent's performance. This project may be expanded in a variety of ways. This model can be modified and trained used to play through all 32 levels. This can be achieved by integrating a meta-learner, we can reduce the time required to train the agent for each level. PPO's generalization will be improved by using a meta-learner. The agent's primary goal in this project is to complete the level by going to the right while avoiding death.

In the future, the addition of other reward functions such as collecting gold and killing foes would be beneficial in creating an RL agent capable of matching the performance of a human player. Aside from that, I want to train the agent using significant computing resources to offer the best and most optimal results.

**References**

[1] Beysolow II, T. (2019). Applied Reinforcement Learning With Python. Berkeley, CA: Apress.

[2] Gottesman, O., Johansson, F., Komorowski, M. et al. Guidelines for reinforcement learning in healthcare. NatMed 25, 16–18 (2019).

[3] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research, 47:253–279, jun 2013.

[4] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning.

[5] Nichol, A., Pfau, V., Hesse, C., Klimov, O. and Schulman, J. (2018). Gotta Learn Fast: A New Benchmark for Generalization in RL.

[6] P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete bayesian reinforcement learning. In ICML, 2006.

[7] R. Hunicke, M. LeBlanc, and R. Zubek. Mda: A formal approach to game design and game research. In AAAI Workshop on Challenges in Game AI, 2004.

[8] Schulman, J., Levine, S., Moritz, P., Jordan, M.I. and Abbeel, P. (2015). Trust Region Policy Optimization.

[9] Wu, Y. and Tian, Y. (2017). Published as a conference paper at ICLR 2017 Training Agent for First-Person Shooter Game with Actor-Critic Curriculum Learning