MIM JOURNAL 2024 illuminate your work
Made in Millersville Journal | www.mimjournal.com

Training AI Agents to Play a Tower Defense Game Using Reinforcement Learning

Mitchell Harrison, Jonathan Rivera, and Justin Stevens

Abstract

This paper presents our efforts in training artificial intelligence (AI) agents to play a tower defense game using reinforcement learning (RL). We developed a tower defense game within the Unity Game Engine and used the Proximal Policy Optimization (PPO) RL algorithm to train the AI. Our goal was to enable the AI to play the game at or above the level of a human player. We encountered challenges from the complexity of our game and the training process. However, we still made noteworthy progress in optimizing the AI's performance.

Introduction

The tower defense genre, such as the games in the Bloons Tower Defense (BTD) series, poses interesting challenges for AI training. An AI in this genre must keep track of a large amount of data related to the game and must make numerous decisions in a brief time span. These decisions include which towers to place, where to place them, and deciding between various tower upgrades. Our project aimed to recreate a tower defense game, BTD, in a video game creation software called Unity and train an AI by allowing it to play the game and rewarding it based on the effectiveness of the decisions it makes. Within the realm of reinforcement learning, there are a variety of algorithms that can be utilized. For our project, we decided to use the PPO algorithm due to its suitability for addressing infrequent reward environments, which are common in tower defense games.

Building A Test Environment for the AI

Figure 1: In game screenshot of our game

In our tower defense game, seen in Figure 1, players strategically place defensive towers which look like monkeys from BTD. These towers are used to defend off waves of enemies called bloons, due to their appearance that mimics a balloon, that follow a path. We chose to randomly generate the map layout for each game session to enhance variability and increase the challenge for the AI. The Unity Game Engine allowed us to develop a dynamic environment for our AI to train in.

Building an Intelligent Agent

Developing an intelligent AI capable of effectively playing our game required the use of three related technologies. First was the integration of the ML-Agents library into our game. ML-Agents enables seamless interaction between the game environment and an RL framework. Importantly, it allows us to create multiple instances of the AI playing the game simultaneously, but in different environments, making the training of the agents much faster.

The second technology we utilized was RL. The key idea of RL is training an agent to make good decisions by rewarding it or punishing it based on an action or series of actions. The AI agent learns to make better decisions over time by choosing actions that have previously given it a reward. Since this process relies heavily on rewards and punishments, developing an effective reward structure is crucial to guiding the learning process for desirable behaviors.

This is where the third technique of using a specific RL algorithm called PPO came into our AI's development. PPO operates by adjusting the AI's decision-making strategies using what is called policy gradients, a form of complex mathematics used with three dimensional or higher data to search for better actions. These adjustments attempt to make the biggest improvement in the agent possible without forgetting previously learned information.

Training the AI

Training the AI agents was a multi-stage process. We began with one AI agent to ensure it was working properly. Once we confirmed that it was working properly, we increased the number of agents to 51, thus allowing the AI to train 51 times faster.. Once this change was made, we began fine-tuning the reward structure and other variables that influence the agents to achieve a desirable result.

Figure 2: Showing off 9 of the 51 agents training at the same time

To further increase the intelligence and efficiency of our AI, we employed various optimization techniques, which included curriculum learning, reward shaping, and policy entropy regularization. Curriculum learning involves gradually increasing the complexity of training scenarios, allowing the AI agent to learn progressively more challenging strategies. This involved changing the number of towers it could choose from and allowing the agent to play on a larger map. Reward shaping involves fine-tuning the strength of each reward or punishment to provide the AI with better feedback. Policy entropy regularization helped prevent our AI becoming stagnant as it promotes continual learning improvement. It encouraged the AI to explore its options and strategies, ensuring that its decision-making remains diverse.

Results

Through multiple rounds of training, encompassing over 250 hours of training, we witnessed substantial progress in the performance of our AI agents. With our continuous improvements to the AI after each training session, the AI was able to reach a further wave each time seen in Figure 3. This progress was theorized to be closely tied to our strategic adjustments, particularly in refining the reward system and initially restricting the AI's interaction with various game mechanics. These modifications not only boosted the AI's overall performance but also significantly enhanced its learning rate.

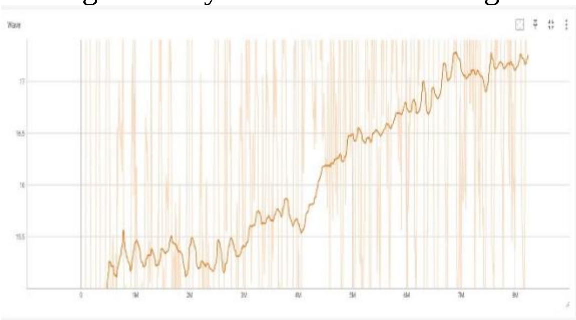

Figure 3: Graph showing that as more actions take place, higher waves are reached

Conclusion

Our research demonstrates the potential of utilizing reinforcement learning to train AI agents for complex gaming environments like tower defense games. Despite the many challenges that we faced, we made noteworthy progress in optimizing AI performance. Although our AI has not surpassed the level of a human, we plan to make more improvements until it does.