IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Intelligent Game Agents using Reinforcement Learning

Yash Nikum[1], Vedant Pawashe[2], Pranav Potdar[3], Harsh Sarda[3], Prof. Nilam Honmane[5]

[1,2,3,4]BE Students, Department of Information Technology, Zeal College of Engineering and Research, Pune, Maharashtra, India

[5]Professor, Department of Information Technology, Zeal College of Engineering and Research, Pune, Maharashtra, India

Abstract:
The area of Reinforcement Learning has witnessed various developments in the last decade. Various RL methods are being researched upon. In this paper, we will employ two of such methods namely Behaviour Cloning (BC) and Curriculum Learning. We will employ these methods to train AI Agents (Bots) to play our Game designed in Unity 3D Engine. By employing using various techniques, we hope to train these AL agents to a human level on this Game. The Game that we showcase is a Platformer, where the player has to get to a finish point somewhere on the map for its completion. We will also design various levels each with growing difficulty. Upon training completion, we will be utilizing the level with the greatest difficulty. Performance of every Agent will be evaluated based on various in game parameters.

Keywords: Reinforcement Learning, Game Agent, Proximal Policy Optimization (PPO), Unity 3D, Behaviour Cloning, Curriculum Learning, Game AI, Generative Adversarial Imitation Learning, AI Agent

I. INTRODUCTION

Reinforcement learning is a novel artificial intelligence paradigm that allows agents to learn by direct interaction with their environments. Through receiving rewards or penalties for their actions, these agents continuously improve their strategies to reach optimal performance. Reinforcement learning has been dramatically fueled by its exceptional achievements in games like Go and Atari, where its potential and variability have been unveiled in the video game field. In this project, we will design an artificial intelligence (AI) agent to learn playing games by itself based on reinforcement learning (RL). While conventional AI systems aid the player or guide the player, we aim at creating an agent that can learn to play games independently based on its exploration of the gameplay complexities, without taking decisions based on its interaction with the environment alone.

II. MOTIVATION

The growing need for intelligent systems across various industries has developed Reinforcement Learning as a vital tool in constructing autonomous agents that are capable of learning to adapt to intricate, dynamic environments. Games are the perfect environments for testing RL algorithms because they are dynamic and interactive. They are a simulation of real-world decision-making situations and offer a manipulated environment where agents can learn and get better. This project entails teaching game agents how to handle difficulties such as stepping over obstacles, ramping, and bridging crossings using optimal RL methods to facilitate improvement over time. This project employs the cutting-edge RL algorithm Proximal Policy Optimization (PPO), which allows agents to trade off exploration and exploitation in a manner that refines strategies. PPO is especially suitable for dealing with high-dimensional worlds and intricate decision-making, making it a good fit for this project.

The project takes advantage of the robust and versatile platform that Unity offers, using it to develop interactive, immersive worlds that will make it easier to train such agents, thereby enabling them to adapt to challenges and learn from each encounter. Although the focus of the project is on designing and enhancing highly competent game agents, its potential applicability transcends into robotics, autonomous mobility, and online training. This work's outcomes will assist in solving real-world automation and intelligent systems problems. At the end of the day, this project is motivated by the need to improve AI technology, expand the limits of what is achievable in intelligent decision-making, and develop systems that can learn and adapt by themselves.

III. LITERATURE REVIEW

3.1 Paper Name: Replicating video game players behavior through deep Reinforcement learning algorithms.

Author: Hafsa Ghabri, Adbelhadi Fennan, Elaachak Lotfi.

Abstract: This work provides a solution for mimicking video game players' behavior through the use of Deep Reinforcement Learning algorithms. Training intelligent agents through algorithms like Proximal Policy Optimization (PPO), Behavioral Cloning (BC), and Generative Adversarial Imitation Learning (GAIL), we provide the agents the capability to learn from their game environment interactions and optimize their actions from rewards and penalties. Experimental assessments on several video games prove that these trained agents are able to effectively emulate human player behavior in challenging situations. This ability presents important opportunities for the generation of challenging non-player characters (NPCs), adaptive difficulty level design, and improved overall gaming experience. Our results indicate that the incorporation of Reinforcement Learning methods enables game developers to offer more realistic and engaging gameplay, successfully closing the gap between Artificial Intelligence and video games.

3.2 Paper Name: A Deep Reinforcement Learning Agent for Snake Game

Author: Md Meem Hossain1, Akinwumi Fakokunde, Omololu Isaac Olaolu

Abstract: After watching *AlphaGo*, a Netflix documentary showcasing how AlphaGo, an AI-powered game developed by DeepMind Technologies, utilizes Deep Reinforcement Learning (DRL), my interest in reinforcement learning has been steadily growing. In this project, I will apply reinforcement learning to design an agent capable of playing the classic Snake game. Deep learning will employ a neural network to assist the agent (snake) in determining the optimal actions required to reach a desirable state. If we define a Deep Reinforcement Learning (DRL) model, the agent interacts with the environment and selects actions. Based on the chosen action, the agent receives feedback from the environment in the form of states (perceptions) and rewards. The state is represented as an 11-input value array, where each input value corresponds to a neural network output consisting of three values. These values represent the three possible actions the agent (snake) can take: moving straight, turning right, or turning left.

3.3 Paper Name: Reinforcement Learning as an Approach to Train Multiplayer First Person Shooter Game Agents.

Author Name: Pedro Almeida, Vítor Carvalho, AlbertoSimões

Abstract: Artificial Intelligence bots are widely employed in multiplayer First-Person Shooter (FPS) games. Using Machine Learning methods, we can enhance their performance and bring them to human levels. In this paper, we emphasize comparing and integrating two Reinforcement Learning training architectures: Curriculum Learning and Behavior Cloning, in an FPS game developed using the Unity Engine. We have created four squads, each consisting of three agents: one squad trained using Curriculum Learning, one using Behavior Cloning, and two employing different approaches to combine both methods. After training, each agent was paired to compete against an agent from another team, with matches continuing until a pairing reached either five victories or ten time-outs. The results showed that agents trained with Curriculum Learning outperformed those trained with Behavior Cloning, achieving 23.67% more average victories in one instance.

Regarding the combination approaches, agents trained with both methods not only faced difficulties during training but also delivered subpar performance in battles, with an average of zero wins.

3.4 Paper Name: Reinforcement learning for obstacle avoidance application in unity ml agents.

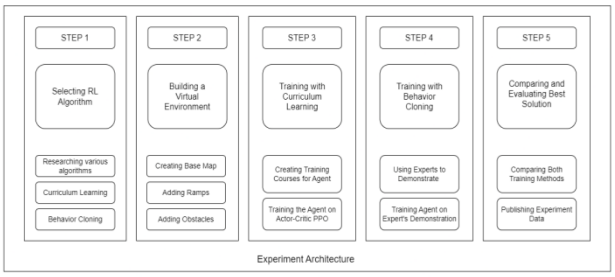
Author name: Reza Mahmoudi and Armantas Ostreika

Abstract: Advances in artificial intelligence have created new opportunities for researchers to address previously challenging use cases. A very good example is simulated autonomous driving automation, which has been a long-standing challenge. But with advances in reinforcement learning (RL), researchers have been able to produce good results. RL's Proximal Policy Optimization (PPO) algorithm was employed to cross-validate models on racecar agents in a Unity environment, as this paper demonstrates. The architecture of ML agents in Unity Engine is especially convenient to experiment with RL algorithms. Behavior cloning is a popular machine learning technique where a model is learned from expert demonstrations. It has been extensively used in many applications like robotics, autonomous vehicles, and video games. Generative Adversarial Imitation Learning, popularly referred to as GAIL, is a form of reinforcement learning that is utilized to learn policies from demonstration data in cases where the action distribution is unknown. GAIL has a generator and discriminator network, which interact in order to acquire a policy which can imitate the behavior of an expert. Agents were trained to understand the environment and drive over the blocks by utilizing the behavior cloning as well as the GAIL algorithms. Different barriers were added to the environment in the experiment, and behavior cloning as a pre-training method and Generative Adversarial Imitation Learning (GAIL) were used together to train agents to navigate through barriers. The best model attained a total reward of -1.619 and a value loss of 0.019 using the behavior cloning method described above with the use of the PPO algorithm.

Sr. No	Title of Paper	Published Year	Methodology	Technology/Algorithm Used	Limitations / Research Gap
1.	**Reinforcement Learning as an Approach to Train Multiplayer First Person Shooter Game Agents.**	**2024**	**- Various RL architectures are tested using an FPS environment. Agents were dueled with each other and result was compared.**	**-PPO -GAIL -Behavior Cloning -Unity ML-Agents**	**The author used FPS games as the environment for training the agents. Author also mentioned the need of implementing this type of algorithms in different kinds of environments.**
2.	**Replicating Video Game Players' Behavior Through Deep Reinforcement Learning Algorithms**	**2024**	**Trained AI agents using GAIL, a type of Behavior Cloning. And evaluated its performance against real-life players.**	**-Proximal Policy Optimization (PPO) -Behavioral Cloning (BC)**	**The Conventional Deep Q-learning performance lacked because it is unable to prevent overfitting problems and because the targets would be the Q-values of each of the actions for training the neural network.**
3.	**A Deep Reinforcement Learning Agent for Snake Game**	**2023**	**- Reinforcement Learning - DNN - Deep Neural Network - DRL - Deep Reinforcement Learning**	**Monte Carlo (MC)**	**The agent does not learn to avoid scrolling or biting itself. It learns to avoid any obstacle in front of the snake head, but it cannot see the whole game. When the snake is longer, the agent will scroll itself and die. In consequence, the highest score of the game is always confined to a certain number.**
4.	**Reinforcement learning for obstacle avoidance application in unity MLAgents**	**2023**	**Mojoco is used to make physics simulation. Agents shaped humanoids are trained using openai gym interface.**	**MOJOCO OPENAI GYM PPO**	**The limitations of this project is that training equipment and data needed is vast and is currently unavailable.**

IV. METHODOLOGY

4.1 System Plan


STEP 1
Selecting RL Algorithm
Researching various algorithms
Curriculum Learning
Behavior Cloning
STEP 2
Building a Virtual Environment
Creating Base Map
Adding Ramps
Adding Obstacles
STEP 3
Training with Curriculum Learning
Creating Training Courses for Agent
Training the Agent on Actor-Critic PPO
STEP 4
Training with Behavior Cloning
Using Experts to Demonstrate
Training Agent on Expert's Demonstration
STEP 5
Comparing and Evaluating Best Solution
Comparing Both Training Methods
Publishing Experiment Data
Experiment Architecture

4.1.1 Problem Formulation

A platformer is a very widespread type of game in which the basic goal is to transport player from one place to another. Arrival at the ending point will demand passing through barriers, challenging terrains. Player has to execute action like jumping or evading traps. Here our intention is to build an agent employing cutting-edge algorithms like Proximal Policy Optimization (PPO), Behavior Cloning (BC) and Generative Adversarial Imitation Learning (GAIL)[3].

4.1.2. Environment Design

A platformer involves the player moving over various obstacles to the endpoint. We will be developing our platformer game based on the Unity 3D engine. Rotating bridges, spiked minefields, rolling balls, and compulsory switches will be included as obstacles. Special hidden rewards might be there, where the player will receive an enormous amount of points but will be hard to reach.

4.1.3. Curriculum Learning Training[14]

Curriculum Learning will be employed for training our first Agent. Agent will be trained on a predetermined curriculum wherein the complexity will increase as we add the levels. We plan to have a minimum of 4+ levels for training the agent at the start.

4.1.4. Behavior Cloning Training[3]

For training the agent with Behavior Cloning we employed recorded Demonstration files with the highest level utilized for training Curriculum Learning. We will carry out Curriculum Learning employing GAIL algorithm.

4.1.5. Testing

To compare the agents with one another we will analyze how many points/ completion time of every agent. The agent who gained the most points or the agent who completed the quickest will be treated as the winner. This test will be carried out on the highest level.

4.1.6 Metrics

Performance of the Agents will be calculated through the parameters attained while carrying out the run. The parameters include completion time, walked distance, success rate for the run, points acquired.

4.1.7 Algorithms:

4.1.7.1 Reinforcement Learning [2]

Reinforcement Learning is a branch of machine learning. It puts an agent in an environment, the agent learns to make the right decisions by interacting with an environment in order to maximize rewards in the long run. RL is composed of different parts like agents, policy, and the reward signals.

- **RL Agents**
An agent is an entity which interacts with the environment. Agent monitors state of the environment and performs actions.

- **RL Environment**
The Environment is the space that agent is deployed in. We will be using Unity 3d for creating this environment.

- **RL Policy**
RL policy defines the mapping between states and actions, indicating what actions the agent must take for the state.

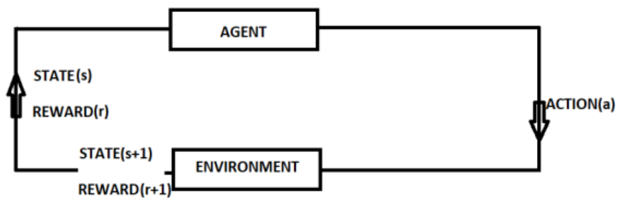


Fig. Reinforcement Learning Process [3]

4.1.7.2 Proximal Policy Optimization (PPO) [15]

Proximal Policy Optimization is a reinforcement learning algorithm developed by OpenAI. It is used to create a policy, a policy in RL is a mapping function which maps action space to state space. In RL, an agent takes actions, receives rewards, and improves its strategy over time. PPO is classified as a policy gradient method for training an agent's policy network. The policy network is the function that the agent uses to make decisions. Essentially, to train the right policy network, PPO takes a small policy update, so the agent can reliably reach the optimal solution.

4.1.7.3 Generative Adversarial Imitation Learning (GAIL)[9]

GAIL algorithm learns a policy by concurrently training alongside a discriminator network, which is tasked to evaluate the policy with respect to the expert demonstrations. These demonstrations files are pre-recorded by a human player playing the game.

V. EXPECTED OUTCOME

During this research, we will use state-of-the-art Reinforcement Learning Algorithms on our platformer game. Through this we want to compare two different approaches to Reinforcement Learning namely Behavior Cloning and Curriculum Learning. Expected Outcome will be two Agents which will be able to play the platformer game as intended and try to grab highest number of points. This all will be measured using different metrics and results will be published. This will help us understand which algorithm was better for our use case.

VI. CONCLUSION

In conclusion, Reinforcement Learning can be performed approaches that one can use to train their own AI Agents. Extensive Research is being performed on RL. In this research we try to understand RL and different approaches of RL namely Behavior Cloning and Curriculum Learning.

VII. REFRENCES

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, "Playing atari with deep reinforcement learning," arXiv preprint arXiv:1312.5602, 2013.

[2] Pedro Almeida, Vítor Carvalho, AlbertoSimões "Reinforcement Learning as an Approach to Train Multiplayer First Person Shooter Game Agents."

[3] Taresh Dewan, Aloukik Aditya, Manva Trivedi Ao Chen " The Use of Reinforcement Learning in Gaming:The Breakout Game Case Study ".

.

[4] McPartland, M.; Gallagher, M. Reinforcement Learning in First Person Shooter Games. IEEE Trans.Comput. Intell. AI Games 2011, 3, 43–56. [CrossRef]

[5] Unity Team. The ML-Agent's Github Page. Available online: https://github.com/Unity-Technologies/ml-agents (accessed on 20 June 2023).

[6] Silver, D.; Huang, A.; Maddison, C.; Guez, A.; Sifre, Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V; Mastering the game of Go with deep neural networks and tree search. Nature 2016,529, 484–489. [CrossRef]

[7] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning:An introduction. MIT press, 2018

[8] Gagniuc, Paul A. Markov chains: from theory to implementation and experimentation. John Wiley & Sons, 2017

[9] K. Arulkumaran, M.P. Deisenroth, M. Brundage and A.A. Bharath, "Deep reinforcement learning: A brief survey," IEEE Signal Processing Magazine, vol. 34, no. 6, pp. 26-38, 2017.

[10] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup and D. Meger, "Deep reinforcement learning that matters," In Proceedings of the AAAI conference on artificial intelligence, vol. 32, no. 1, April 2018

[11] C. Szepesvári, "Algorithms for reinforcement learning," Synthesis lectures on artificial intelligence and machine learning, vol. 4, no. 1, pp. 1-103, 2010.

[12] D.J. Soemers, C.F. Sironi, T. Schuster and M.H. Winands, "Enhancements for real-time Monte-Carlo tree search in general video game playing," In 2016 IEEE Conference on Computational Intelligence and Games (CIG), pp. 1-8, September 2016.

[13] Z. Wei, D. Wang, M. Zhang, A.H. Tan, C. Miao and Y. Zhou, "Autonomous agents in snake game via deep reinforcement learning," In 2018 IEEE International conference on Agents (ICA), pp. 20-25, July 2018.

[14] Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In Proceedings of the 12th International Conference on Machine Learning (ICML 1995), pages 30–37. Morgan Kaufmann, 1995.

[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv.org, https://arxiv.org/abs/1707.06347, arXiv:1707.06347[cs.LG].