Reinforcement Learning in EVE Online

Pranav Hegde **Shareysan Ravishankar**

Danylo Voloshyn **Nikhil Chandrasekar** **Jonathan Yang**

Abstract

Developing autonomous agents in a dynamic online environment is a long-standing challenge for reinforcement learning algorithms. In this paper, we explore our approach to using Reinforcement Learning to design an autonomous mining bot for the space MMORPG game EVE Online, leveraging OpenAI's Gymnasium's RL training API to construct a simulated environment. The bot is trained to perform optimal resource extraction in a time efficient manner while maintaining the survival needs of the player, like buying fuel, in the cheapest and most non-invasive way possible. We defined this task as a Markov Decision Process and utilized deep reinforcement learning techniques, primarily Deep-Q Networks and varieties of Proximal Policy Optimization (PPO), to train the agent to intelligently complete the proposed task. Our results show that the agent is capable of adapting to its environment and optimizing its effectiveness by exhibiting behaviors like mining the most valuable minerals and selling them in locations that have the highest prices. When comparing the effectiveness of the bot, which we quantified by the amount of in-game currency it earned per episode, we saw a gradual but unmistakable increase in profit over training episodes, and therefore a corresponding increase in overall efficacy. This project showcases the feasibility of utilizing RL with the Gymnasium-simulation framework to develop intelligent and adaptive agents capable of handling an ever-changing ecosystem.

1 Introduction

1.1 Motivation and Context

Teaching machines how to play games has held an important role in the history of artificial intelligence. Early efforts relied on manually defined heuristics and search methods, exemplified by systems such as IBM's Deep Blue, which defeated Garry Kasparov in Chess in 1997. With the rise of Reinforcement Learning (RL), machine learning practitioners have moved beyond these rule-based and fully observable environments towards environments with richer dynamics and partially observable states. Breakthroughs in deep learning, such as Google DeepMind's Deep Q Network (DQN) outperforming humans on various Atari games and AlphaGo's victory over Lee Sedol in 2015 showed the power of combining traditional reinforcement learning techniques with deep learning methods [1][2]. Additionally, more recent successes such as AlphaStar in StarCraft II demonstrated that RL agents can navigate complex real-time strategy environments with large observation and action spaces, partial observability, and uncertainty. Building on this progress in previous works, our focus shifts to EVE Online, an expansive open-world MMORPG that poses novel challenges and could provide potential breakthroughs in areas with real world applications. For instance, resource allocation, strategic decision-making, logistics operation, fleet combat and multi-agent coordination among dozens of other gameplay mechanics that mimic real world economic and logistical environments. By demonstrating the development of effective RL agents that can perform a variety of tasks in EVE

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

Online's environment, this project hopes to build a path for versatile and generalizable reinforcement learning solutions that can be deployed in similar real world domains.

1.2 Project Goals

The overarching goal of this project is to develop and evaluate a reinforcement learning agent capable of operating in EVE Online or a simulated EVE Online environment. Specifically, we aim to:

1. **Explore Game Interaction:** Explore various methods to interact with game states directly and attempt to build an RL agent that can function with this interface

2. **Customize Simulation Environment:** Implement a scalable, Gymnasium-based environment that captures core gameplay mechanics of EVE Online for relevant tasks, such as resource distribution, pricing variation across the in-game universe, energy/fuel constraints, and cargo restrictions

3. **Experiment with RL Agent Architecture:** Implement and evaluate various RL techniques, particularly Deep Q Networks (DQNs) and Proximal Policy Optimization (PPO), in the agent

4. **Demonstrate Adaptive Behavior:** Show that the agent can learn and adapt novel strategies over time and respond to changing environments without human guidance

Our team presents a single-agent reinforcement learning system for playing the space-based MMORPG game EVE Online. Specifically, our agent is trained, tested, and deployed on a novel, self-contained simulation of the game which rapidly accelerates training and enables the aggregation and subsequent observation of a continuum of game states. We document here the design, construction, and implementation of an emulated game environment, and describe certain unexpected behavior observed during the training of the agent within. We shall demonstrate the full capabilities of the agent, which meet the primary goal stated in our Project Proposal, and speculate on its direction and future development given more time.

2 Related Works

Reinforcement learning has been explored in the realm of gaming across many avenues, including League of Legends, a 2009 multiplayer online battle arena game. Using this environment, one paper explores the use of Proximal Policy Optimization (PPO) involved in a model to increase the performance of an agent in a custom 1v1 [5]. PPO is an algorithm that directly optimizes a policy, introducing a clipping mechanism to ensure that the new policy is not too far off from the old policy while training occurs. The paper noted that the model implementing this algorithm was able to outperform the model that did not include it [5]. StarCraft II, a multiplayer real-time strategy game released in 2010, was another avenue in which deep RL was explored. A 2022 paper delves into the effectiveness of PPO, in which StarCraft II was used to show that despite being implemented significantly less than off-policy algorithms, PPO achieves results equal to or better than the other algorithms [6].

Another algorithm that has been used for reinforcement learning in gaming is the Deep Q-Network (DQN). This algorithm incorporates Q-learning, which aims to find the optimal policy that will maximize the reward seen cumulatively using the Q-function. DQN combines Q-learning with deep neural networks, allowing it to handle large and continuous state spaces. Because of the increased complexity in games over the years, techniques for learning that could adapt to more complex environments are needed, and a DQN is often an appropriate solution. One of the early papers from 2013 introduces the implementation of DQNs for seven popular Atari games and shows that these networks outperformed other learning methods by a significant margin on all seven games [7]. The other learning methods incorporated substantial prior knowledge about the problem and its input in contrast to DQN, which further shows how much more effective it was [7]. Other applications of these networks include StarCraft II agents which were comparable to a novice player on its mini-games [8], and agents in a popular Asian card game named Dou Di Zhu which were comparable to human players [9].

3 Environment Design

3.1 The Curse of EVE Online

3.1.1 EVE Online State Space Complexity

EVE Online presents an intricate environment, characterized by its dynamic player-driven economy and expansive science fiction open world setting. As a persistent MMORPG, EVE Online simulates a massive universe where thousands of live players interact in real-time performing behavior such as mining resources, trading items, and solo and team-based combat. The game's economy is entirely player-driven with in-game item exchanges running on actual central limit order books with in-game supply and demand affecting actual prices, which makes EVE Online an incredibly exciting environment within which to build. The sheer scale of possible interactions and game states creates a complex state space which can be difficult to approximate even for cutting edge RL techniques. Additionally, EVE Online is real-time and online-only, meaning tasks such as warping, mining, and combat require time on the order of seconds or minutes. This conflicts with the fast iteration speed required by reinforcement learning models that train for hundreds or thousands of steps.

3.1.2 Game Client Interaction

Directly extracting data from and interacting with the EVE Online game client proved more difficult than expected. Through initial research, the team identified several promising open source projects that aimed to solve the problem of programmatic access to EVE Online's game client for the purpose of building automated bots or data analysis programs. In particular, we identified "eve-memory-reader," which provided a direct access API for the memory buffers used by the EVE Online game client [2]. Our initial roadmap relied on using the memory reader as a simple interface to retrieve game data. For instance, locations in the player's solar system, nearby players/enemies, asteroids, etc., as well as bounding boxes for interactions such as managing mining equipment, targeting asteroids/enemies/players, firing weapons, and so on. Ideally, our game interaction infrastructure would have looked like the following:

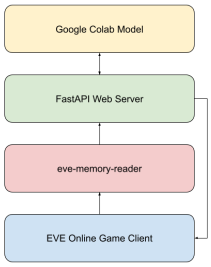


Figure 1: Interaction Infrastructure

In this ideal infrastructure implementation, eve-memory-reader provides a read-only interface to the game client such that the FastAPI Web Server maintains a stream of game state updates and model actions. These are forwarded to the model inference or training running in Google Colab or executed in the game client via the game's memory. Unfortunately, the memory reader is over a year out of date and written in low level C that was difficult to parse. Another issue is that EVE requires Windows 64-bit, which many of the team members did not use on their home machines. We attempted to use the "EVE-Online-Bot" project as a replacement for "eve-memory-reader" as a game client interface, but this project is plagued with additional issues [2][3]. It is a complex interface, relying on several computer vision models in tandem and specific window configurations and screen resolution.

3.1.3 Pivoting to Environment Simulation

EVE Online is often called a "spreadsheet" game, and even has official Excel integration [10]. This means that the vast majority of the visuals are not required for gameplay, and decisions in the game are based on lots of numerical data. This makes it relatively easy to simulate when compared to other genres of video games such as First-Person Shooters (FPSs) that rely on 3-dimensional spatial

awareness and reaction time at the core. Instead of building complex low level software outside the scope of the project to interact with EVE, we opted to build a proxy simulation for the environment.

3.2 Proxy Environment Simulation

3.2.1 Simple Discrete Mining Environment

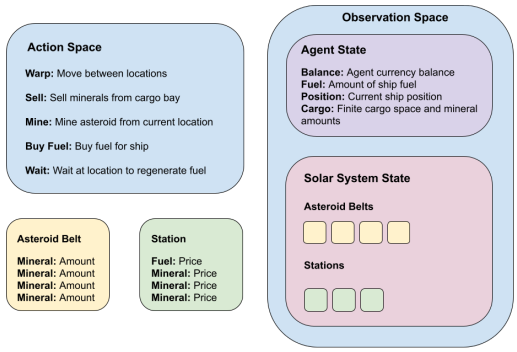


Figure 2: Action and Observation Space

Action	Condition	Reward
Warp	Sufficient Energy	0
Warp	Insufficient Energy	-0.1
Mine	At belt, available cargo space	mined_value * reward_scale
Mine	Not at belt / no cargo capacity	-0.01
Sell	At station, minerals in cargo	sale_revenue * reward_scale
Sell	No minerals in cargo	-0.005
Buy Fuel	Fuel at station, cash balance	0
Buy Fuel	No fuel or low cash	-0.005 or -0.01
Wait	Energy regeneration allowed	-0.001

Table 1: Reward Configuration

This simple environment was implemented as a Gymnasium (formerly OpenAI Gym) class generated from a declarative object describing features of the training environment such as the asteroid belts and their resources as well as stations and their prices [4]. In every generated environment the agent is described by a set of both continuous and discrete values which includes the agent's amount of fuel, cargo, currency balance, and current position. The agent's actions space consists of 5 actions which can be executed at the agent's current location given a set of conditions is met. Additionally, some actions, including warping and mining, cost fuel to perform, forcing the agent to make better decisions concerning its resource management. This configurability allows the generation of environments with varying resource distributions and pricing dynamics between stations in the solar system, providing an environment that can be configured for increasing complexity for testing the limits of various RL methods.

During environment fine-tuning, care was taken to ensure that rewards were balanced properly. During construction of the environment, actions were defined sequentially and their rewards varied relative to one another carefully through grid search. Some assumptions were made on relative importance; for example, being unable to warp to a location to sell off mined ores would equate to a total failure in profits for that specific run, and thus failure to warp would be weighted much heavier than anything related to mining by necessity. We also ensured that inefficient means of 'reward farming' like waiting for passive energy regeneration were punished to boost the agent's overall performance.

4 Models and Experiments

4.1 Deep Q Network (DQN) Model

Using an implementation of the environment defined in section 3.2.1, we implemented a Deep Q Network to attempt to solve the problem of maximizing profit in the face of the constraints defined by the environment. This architecture was chosen due to its history as a performant reinforcement learning model for game environments where the action space consists of a single discrete dimension, such as the environment in section 3.2.1.

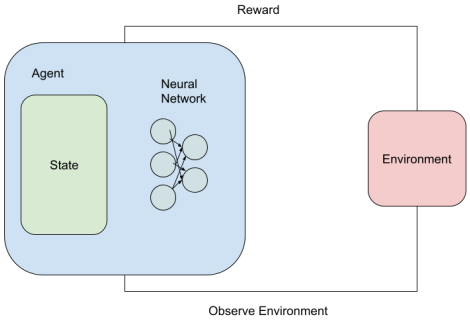


Figure 3: DQN Architecture

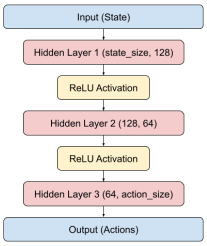
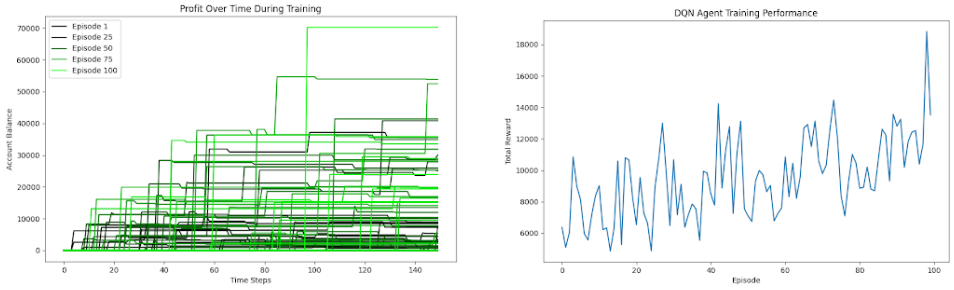


Figure 4: DQN Neural Network Architecture

This Deep Q Network consists of 3 fully connected layers, the first two of which were passed through ReLU activation functions. We trained this DQN agent over 100 episodes consisting of 500 steps each and evaluated the agent against it. Below are the training results:



(a) Profit Over Time (b) DGN Agent Training

Figure 5: Training Images

During the training process, it can be seen that the account balance steadily increases save for the natural volatility of reinforcement learning systems. However, upon evaluating the trained policy on a test environment, the policy was unable to generalize to a new environment and could not make optimal moves. We suspect that the reason for the DQN's struggle with generalization lies in the DQN's inability to handle large observation spaces due to the curse of dimensionality.

4.2 Actor-Critic Proximal Policy Optimization (PPO with Actor-Critic

Using another implementation of the environment defined in section 3.2.1, we implemented an Actor-Critic PPO model to attempt to solve the problem of maximizing profit in the face of the constraints defined by the environment. As opposed to the Deep Q Network technique, Actor-Critic PPO is much more suited for a complex state space such as that of EVE Online. This is due to its use of clipping which ensures stability during training by preventing large updates to the policy. Additionally, by bifurcating the network into one network that learns optimal state -> action mappings, and one that approximates the value of those state -> action mapping allows for Actor-Critic models to converge faster to complex action and state spaces than the Deep Q Network which only produces value estimates for actions.

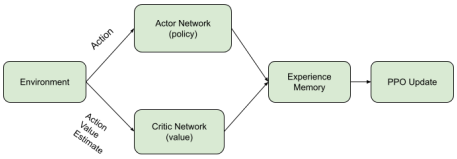


Figure 6: Actor Critic PPO Architecture

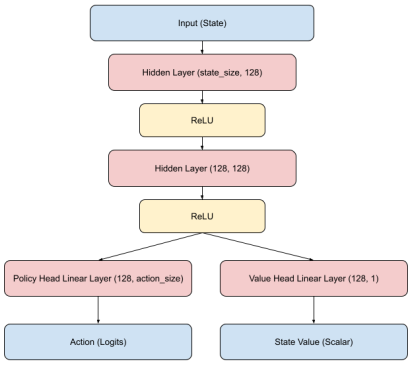


Figure 7: Experimentation With Actor-Critic PPO Technique

4.2.1 Experimentation With Actor-Critic PPO Technique

To train this Actor-Critic PPO model, we leveraged the configurability built into the proxy simulation we constructed to approximate the EVE Online game environment to construct four separate environments of increasing complexity:

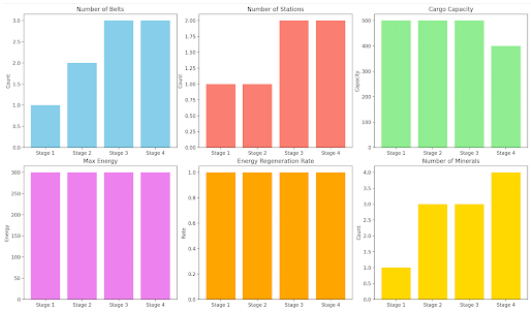
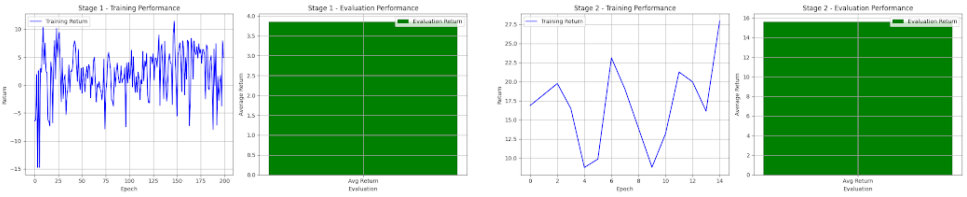


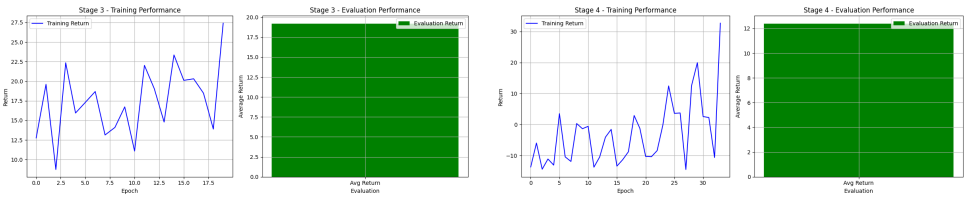
Figure 8: Training Statistics

We then trained a separate Action-Critic PPO model for each scenario in order to see how the model would interact with varying levels of state space complexity. For evaluation, we evaluate the trained

policy against the environment over 100 episodes and plot the average return. As can be seen in the figure below, the Agent Critic PPO model was able to show increasing performance over training episodes even as complexity increased, and was able to obtain a proportional return in evaluation regimen as well, indicating adaptive behavior.



(a) Stage 1 (b) Stage 2



(c) Stage 3 (d) Stage 4

Figure 9: PPO Training and Evaluation

In addition to training a model exclusively on a stage and comparing performance across stages, we also implemented a curriculum learning method, training a single policy incrementally over all stages with an early-stopping mechanism to prevent overfitting on any single stage. However, this method did not lead to higher performance across later stages likely due to early stopping preventing generalization during each stage.

Figure 10: Curriculum Learning

5 Future Work and Conclusion

While our current reinforcement learning agent has shown promising results, there are more opportunities for improvement and implementation. One of the most promising directions for future implementation is the development of a multi-agent system. As of right now, it is just a single-agent system operating on mining, but opening room for more agents will provide more interesting results and more paths to explore. With two more weeks, there could possibly be additional agents with the same framework that are added and tasked with collaborating with one another. Mining efficiently and maximizing resources in a team setting could be observed and analyzed. For instance, agents mining together could find more efficient ways to mine the whole simulated solar system than a single agent. With two more months, having teams of agents that collaborate and compete against one another would be an intriguing addition. There could either be a resource goal that is set or a time

span given to compare how teams perform against each other. Other tasks such as combat between agents and the environment (NPCs) could also be assessed in this timeframe.

Given additional time beyond a few months, changes such as additional frameworks for agents, more complexity in the simulation, and variety amongst implementation could be considered. Implementing something like player versus player combat would be challenging, but also insightful into how agents strategize to take one another down. Different learning processes such as imitation learning, where agents learn from experts, or different policy gradient methods in a MARL system could be explored to see which ones can do better in this type of environment. The teams of agents could be set up unevenly to simulate unfairness, which could provide insights into real scenarios in the game.

References

[1] Clune, Jeff, et al. "Learning to Play Minecraft with Video Pretraining." OpenAI, 23 June 2022, openai.com/index/vpt.

[2] jamesalbert. (n.d.). jamesalbert/eve-memory-reader: Run eve online bots. GitHub. https://github.com/jamesalbert/eve-memory-reader

[3] darkmatter2222. (n.d.). darkmatter2222/Eve-Online-Bot: A bot to play eve online leveraging various methods including popular machine learning tools. at this point, its a full blown AI. GitHub. https://github.com/darkmatter2222/EVE-Online-Bot

[4] Gymnasium documentation. (n.d.). https://gymnasium.farama.org/

[5] Lohokare, Shah, et al. "Deep Learning Bot for League of Legends." 2020, https://doi.org/10.1609/aiide.v16i1.7449

[6] Yu, Chao, et al. "The Surprising Effectiveness of PPO in Cooperative Multi ...", Nov. 2022, arxiv.org/pdf/2103.01955.

[7] Mnih, Volodymyr, et al. "Playing Atari with Deep Reinforcement Learning." arXiv.Org, 19 Dec. 2013, arxiv.org/pdf/1312.5602.

[8] Vinyals, Oriol, et al. "StarCraft II: A New Challenge for Reinforcement Learning." arXiv.Org, 16 Aug. 2017, arxiv.org/pdf/1708.04782.

[9] You, Yang, et al. "Combinational Q-Learning for Dou Di Zhu." arXiv.Org, 19 Feb. 2019, arxiv.org/pdf/1901.08925.

[10] Gross, Chris. "EVE Online and Data Types Apis." techcommunity.microsoft.com, 6 May 2022, techcommunity.microsoft.com/blog/excelblog/data-types-in-space-a-new-data-types-experience/3328453.