League of Legends AI agent with PPO

Diogo Terrantez
University of Porto
Porto, Portugal
up202308526@edu.fe.up.pt

ABSTRACT

This paper explores the development and evaluation of an AI agent for League of Legends (LoL), trained using Proximal Policy Optimization (PPO) from Stable Baselines 3 and interfaced with the game environment through Gymnasium, built atop LeagueSandbox. The study aims to understand the AI's learning interaction within the LoL environment in simplified settings, addressing the challenge of strategic thinking and adaptation in complex scenarios. Results highlighted limitations in training due to LeagueSandbox's unfinished state, impacting training effectiveness, and the weakness of PPO in adapting to the game's extensive action space. Despite modest outcomes, the research underscores the complexity of applying AI in dynamic game environments and suggests future exploration with the official game engine and advanced learning techniques beyond PPO.

KEYWORDS

League of Legends, Gymnasium, Stable baselines, AI Agent

ACM Reference Format:
Diogo Terrantez. 2024. League of Legends AI agent with PPO. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 4 pages.

1 INTRODUCTION

League of Legends (LOL) is a competitive online game that blends the speed and intensity of a real-time strategy with role-playing elements. Two teams of five players compete to destroy the opposing team's base while defending their own, each controlling one of 167 characters known as a "champion", each with unique abilities. The game emphasizes teamwork, strategy, and player skill, making it one of the most popular and enduring esports titles globally. LOL has a lot of potential as an AI learning environment, as it requires strategic thinking, planning, and adaptation to rapidly changing conditions, which can help develop sophisticated AI capable of decision-making in uncertain and complex scenarios. The game is played in a highly complex environment, with a discrete but vast action space, in a partially observable space that requires predicting opponent movements for effective play.

This project aims to understand how an AI agent learns to interact with the LOL environment and, hopefully, train it to play the game in a simplified setting effectively. The agent is trained using the PPO implementation from Stable-Baselines3[1] and Gymnasium[2] to interface with the game environment, built on top of LeagueSandbox[3].

1.1 Stable Baselines 3 and Proximal Policy Optimization (PPO)

Stable Baselines 3 offers an updated suite of high-quality implementations of reinforcement learning algorithms, focusing on ease of use, reproducibility, and efficiency. Among its offerings, the Proximal Policy Optimization (PPO) algorithm stands out for its robust performance across various environments, balancing the trade-off between exploration and exploitation. This makes PPO a preferred choice for developing advanced AI agents, including those designed for complex strategic games like League of Legends.

1.2 Gymnasium for Environment Management

Gymnasium provides a standardized interface for various simulation environments, facilitating the development and benchmarking of reinforcement learning algorithms. Its utility in creating custom environments, such as those mimicking the intricate dynamics of League of Legends, enables AI agents to train under controlled, reproducible conditions.

1.3 LeagueSandbox

LeagueSandbox represents a pivotal development in the customization and simulation of League of Legends gameplay for research and development purposes. It is a community-driven project that replicates the LOL game engine. By offering a controllable and modifiable League of Legends environment, LeagueSandbox enables researchers to simulate specific game states, test various strategies, and refine the decision-making capabilities of AI agents in a controlled setting. This unique environment is essential for developing sophisticated AI models tailored to the complexities of League of Legends.

However, this project infringed on the game's copyright and was shut down in 2022, leaving it in an unfinished and only partially working state.

Gymnasium serves as the interface between the AI agent and the LeagueSandbox engine and allows for direct communication and control of the game's mechanics, enabling the AI to interact with, learn from, and make decisions within the simulated League of Legends environment.

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand.

[1] stable-baselines3.readthedocs.io
[2] gymnasium.farama.org
[3] github.com/LeagueSandbox

2 RELATED WORK

AI research in LOL has been very limited as no API or Sandbox mode exists that would provide an agent direct access to the game engine.

The most well-known research in the area is from OpenAI [1], which created a team of AI agents (OpenAI Five) that defeated the world champions in DOTA 2, another game very similar to League of Legends. OpenAI Five used a Proximal Policy Optimization (PPO) [4] with Long Short Term Memory (LSTM) [2] approach, which was continually trained in a distributed framework consisting of 128,000 CPU cores and 256 P100 GPUs on GCP for ten months.

More recently, a similar approach was attempted for LOL [3]. The group trained a CNN to perform object recognition during a LOL match, using its outputs to train two AI agents (in a simplified game mode), one using LSTM and the other LSTM + PPO. The LSTMs were trained with data from two advanced players and played against a bot built into the game. The models were trained for 72 hours on a single K80 GPU, with the LSTTM + PPO achieving better results.

3 METHOD

The implementation of this project is based on an adaptation[4] of LeagueSandbox that accepts actions and returns observations adequate for a reinforcement learning setting. A further adaptation [5] exists that provides a custom Gym (earlier version of Gymnasium) environment. This project uses this final adaptation as a base for implementation.

3.1 Environment preparation

The Gym environment provided is not compatible with the latest versions of Gymnasium. For that reason, it is used only as a base to interact with the game engine, with a new, Gymnasium-compliant environment built to provide the observations and receive the actions from the agent.

The agent will be learning a very simplified version of the game. It consists of a one-versus-one scenario where it and the opponent play the same champion, "Ezreal". An episode ends when one of the champions dies, and for each step, the agent can only perform one action. The environment operates at seven frames per second, each stem consisting of one frame.

Three versions of this scenario were defined:

Run Away - The agent must distance itself as much as possible from the opponent in a limited number of steps.

Targeting - The must learn to use its abilities and target a stationary opponent enough times to kill it in a limited number of steps.

1v1 - The agent will play against itself, learning to dodge the opponent's abilities, automatically targeted at its last known position.

3.2 Run Away

This is the simplest scenario, in which the agent is only allowed to move. The environment works with "8-directional movement", typical of digital games in 2D spaces, where the agent can choose a discrete value from zero to seven for the X and Y components.

The action space consists of those two values, while the observation space is the Euclidean distance between the two champions.

In this scenario, the agent and stationary opponent are teleported near the center of the map, where the agent must distance itself as much as possible for the episode duration.

3.3 Targeting

The agent can perform two actions in this scenario: a movement input like the previous scenario or an ability.

While the champion has multiple abilities, for simplicity, the agent is only allowed to use one, a directional projectile that launches from the character to a point in space. After using the ability, the agent cannot use it again for a set amount of time.

The action space now has five discrete elements: a boolean to decide whether to move or use the ability, the movement components from before, and two new X and Y components to be used as the ability target position. These values range from 0 to 15000, which is the size of the entire environment map.

The observation space is also more complex, now having discrete values for the agent character player position (X and Y), health points of the champion, and ability cooldown. This is duplicated for the opponent champion, and the distance to the opponent is maintained from before.

In this scenario, the agent and stationary opponent are also teleported near the center of the map, where the agent must target the opponent with its ability and kill it within the episode duration.

3.4 1v1

This is the most complex scenario, where the agent will play against another instance of itself, which controls the opponent character. The action space is simplified, only having the movement component from the first scenario and the boolean to decide between a movement or ability action. The observation space now also contains a copy of the previous action performed by the agents.

When the agent decides to use an ability, it is automatically targeted at the enemy position from the previous observation.

The objective in this scenario is for the agent to learn how to dodge the opponent's ability while using its own.

3.5 Agent definition

The AI agent in this project is an implementation of PPO from the Stable Baselines 3 Python library. It is built to be compatible with Gymnasium environments, having access to its action and observation spaces, and able to automatically advance the environment while learning.

Stable Baselines allows control of various hyperparameters (e.g., learning rate, policy arguments, etc.), but they are kept as default for this project.

4 EXPERIMENT AND RESULTS

For each scenario, a new PPO agent was created (with the same parameters). This is because Stable Baselines does not allow an agent to work with multiple action spaces. This means training starts from scratch for every experiment. Model performance during training is measured by its mean reward and subjective observation of its behavior in the environment.

[4] github.com/MiscellaneousStuff/pylol
[5] github.com/MiscellaneousStuff/lolgym

The models are trained until training is stabilized, and reward mechanisms are changed when needed. The PPO implementation from Stable Baselines trains in a batch size of 2048 steps (across all episodes). For the experiments, it is trained in multiples of five batches (10240 steps) and saved at the end, marking an epoch.

4.1 Run Away

This scenario is very simple, and the agent was able to solve it within the first training loop. each episode runs for 25 steps, with the reward given at the end of an episode being the Euclidean distance between both characters. This experiment was conducted for only one epoch, as the agent learned to distance itself from the opponent until the episode ended continuously.

4.2 Targeting

For this experiment, the reward was split into multiple components:
Aim reward - The agent is rewarded to point its ability close to the opponent's position. It follows an exponential function split for the X and Y components, each maxing at the opponent's exact coordinate.
Damage reward - The agent is rewarded by damaging the opponent with the spell, effectively rewarding it more for using its ability and aiming near the target.
Ability use - Penalizes the agent for attempting to use the ability when in cooldown, effectively balancing its movement and ability actions.
Distance to opponent - Penalizes the opponent for moving too far from the opponent. This limits the model exploration to the relevant part of the experiment, quickening the training process.

Each episode in the experiment lasts for 1000 steps, and the agent cannot use the ability for the next ten steps for each use. The experiment ran for 1.5 million steps but stabilized around the 200k mark, with the best performance around the half-a-million mark.

Its performance, however, is not as expected. While the training has stabilized, the agent has not learned how to target the opponent's location, even if it stays still. The agent will use its ability randomly, and the deterministic action is to walk in a direction without using its ability.

4.3 1v1

The final experiment has similar reward components to the previous one. The agent is rewarded for damaging the opponent but also penalized for taking damage (1.1x multiplier when damaging). It is still penalized for attempting to use the ability when on cooldown and must also maintain a maximum and now minimum distance to the opponent. This final component incentivizes the agent to keep an optimal distance to dodge the opponent's ability.

The experiment ran for almost one million steps, also stabilizing near the 200k mark, but this time maintaining a much closer reward mean, achieving a peak around the 3000k steps.

It is difficult to evaluate this experiment as ability targeting is automatic, and random movements can also lead to dodging. The agent learning is also limited since, by playing with itself, the dodging ability will be equal, and the agent will stop dodging if it stops hitting abilities and stops using abilities when not able to dodge.

However, from observing the model at different training stages, its behavior changes over time. As training progresses, it begins to move sideways, optimal to dodge directional abilities. In earlier stages, with the reward weighted to penalize the agent for taking damage over damaging the opponent, the agents would stop using abilities altogether.

Figure 1 shows the mean reward for the agent in the Targeting (blue) and 1V1 (violet) scenarios

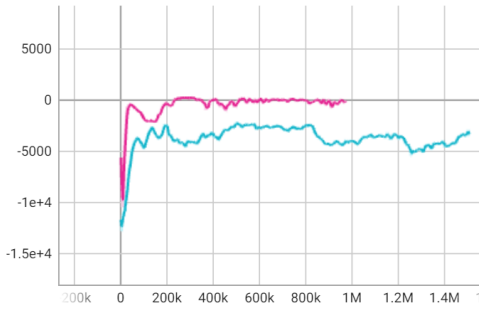

Figure 1: Agents training progress

Some videos [6] are also available to showcase the agent performance.

5 CONCLUSIONS

This project ended with unexpectedly weak results. The game engine seemed promising initially, but its unfinished state caused many problems during implementation and training. Object collision was not implemented, resulting in characters walking through walls at times, instead of around them. A step multiplier option was also available, which should speed up the game engine for faster training. However, this was not implemented correctly, and the engine could not keep up, resulting in abilities going through the champions instead of hitting them. For this reason, I could not speed up training, taking a lot longer for each trial.

The PPO performance was also poor in the targeting experiment. Multiple approaches were used in that experiment, with split and joint observations for position and targeting and multiple reward mechanisms for the aiming component. Still, the agent was unable to learn targeting in such a wide action space. Because of that, I had to resort to auto-targeting for the final experiment and could not reach a point where the agent could learn to predict the opponent's movement for its targeting.

Finally, Stable Baselines does not support multi-agent training in the same environment, so I had to resort to having a copy of the agent in the Gymnasium environment to control the opposing champion, again hindering the training process.

For future work, I will use an approach similar to the related work, using the official game engine and a CNN to receive the observations and control the champion with keyboard and mouse inputs. Having a built-in opponent in the game is also beneficial for training. Using only PPO is also not an effective approach for a game of such complexity and would not be viable for experiments

[6] Showcase videos

more complex than those conducted in this project. Some transfer learning approaches from Human gameplay would also be an interesting concept to explore.

REFERENCES

[1] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. 2019. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680* (2019).

[2] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[3] Aishwarya Lohokare, Aayush Shah, and Michael Zyda. 2020. Deep learning bot for league of legends. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Vol. 16. 322–324.

[4] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).