AI Squared Tournament: A Flexible Reinforcement Learning Framework for 1v1 Platform Fighting Agents

Kaden Seto
University of Toronto
kaden.seto@mail.utoronto.ca

Doga Baskan
University of Toronto
doga.baskan@mail.utoronto.ca

Martin Tin
University of Toronto
martin.tin@mail.utoronto.ca

Steven Lin
University of Toronto
yucheng.lin@mail.utoronto.ca

Zain Moustafa
University of Toronto
zain.moustafa@mail.utoronto.ca

Ambrose Ling
University of Toronto
ambrose.ling@mail.utoronto.ca

Asad Khan
University of Toronto
asadk.khan@mail.utoronto.ca

Matthew Tamura
University of Toronto
matthew.tamura@mail.utoronto.ca

Andrew Magnuson
University of Toronto
andrew.magnuson@mail.utoronto.ca

***Abstract*—Reinforcement Learning (RL) is an often overlooked area of Machine Learning, resulting in the number of opportunities for people to learn the subject oftentimes being limited. The goal of AI Squared is to create a way that allows for people of all backgrounds to learn RL in a fun, competitive, and exciting way. The AI Squared Project consisted of a tournament and a structured iPynb Notebook to allow people to design, train, and battle AI agents, teaching them RL concepts along the way. In the tournament, agents fight in a custom environment, a 1v1 knockout fighting game inspired by Brawlhalla.**

I. Introduction

A. Motivation

Reinforcement Learning (RL), compared to other domains of Machine Learning (ML), is an often overlooked area of ML, resulting in the number of opportunities for people to learn the subject, be it in academic institutions or through online courses, being comparatively limited. At the University of Toronto for example, the only opportunity for undergraduate students to learn RL is in the course ECE411 Adaptive Control and Reinforcement Learning, a course reserved mostly for fourth year Electrical & Computer Engineering students making RL inaccessible for most students studying Computer Science, Data Science or other forms of Engineering [1]. Beyond just a lack of avenues from academic institutions, it is difficult for students to gain practical experience with RL due to the implemented algorithms in public codebases tending to be complex. As Moerland et. al identified, publicly accessible RL test environments are either high-dimensional making experimentation difficult, slow and resource intensive, while the available low-dimensional RL environments such as CartPole and MountainCar fail to capture the full range of possibilities that RL can do [2, 3].

Thus, there is an opportunity in RL education to better provide those interested in the field with accessible practical opportunities to engage, experiment, and ultimately learn RL. Based on the state of RL education at the University of Toronto and the broader field as a whole we wanted to create a solution that could: 1) provide people with hands on-experience with RL in way allowing them to learn, 2) create a solution and way of distributing that is lightweight and easily accessible, 3) make it experience invariant so beginners and students more versed in RL theory can engage, and 4) something overall engaging encouraging people to use.

B. Related Works

RL Environments for Games - There are a number of existing standardized RL environments that allow people to conduct RL research and learn. The most notable of them is OpenAI Gym, which introduced a collection of benchmark game environments, including CartPole, MountainCar, and LunarLander, which have become widely used for testing RL algorithms [3]. Beyond classic toy problems, there also exists RL environments designed for training agents within pre-existing games such as ViZDoom that provides an RL environment to develop and build AI agents to play the video game Doom, and Gym-Retro which extends the OpenAI framework allowing agents to develop policies for games like Street Fighter II and Sonic the Hedgehog [4, 5]. Learners may find it easier intuitively relating actions to reward function design if dealing with a game environment they are more familiar with. Given that most participants were undergraduate university students from the University of Toronto, creating a custom environment with a game more popular among this demographic, Brawlhalla, was done instead.

RL For Education - Recent efforts have explored the use of RL in educational settings to enhance accessibility and engagement. EduGym provides an interactive notebook-based approach to RL education, offering structured exercises that introduce RL concepts through hands-on coding experiences. This approach lowers the entry barrier for students by providing pre-configured environments and curated problem sets [2]. Another notable initiative is the augmented reality-based RL learning platform developed by Zhang et al., which introduces RL concepts to K-12 students through physical robots. By integrating teaching with tangible robotic tasks, this approach improves user engagement and provides an intuitive understanding of RL [6].

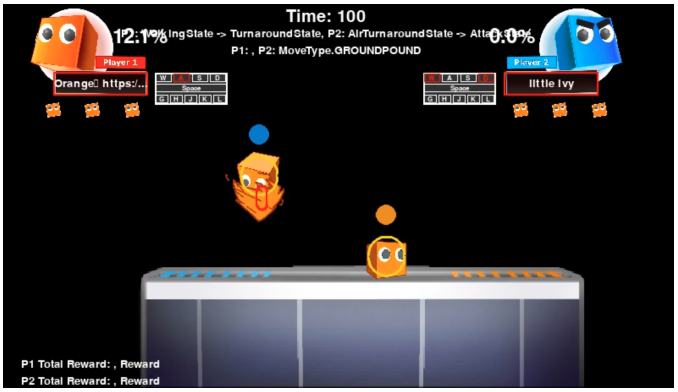

Fig. 1: AI Squared Game environment: 1v1 fighting game with two AI agents in a knockout battle environment inspired by Brawlhalla

C. Problem Definition

Current RL environments and educational tools fail to provide an engaging, competitive, and accessible way for students to train agents in a real-world-inspired game setting. While existing frameworks like OpenAI Gym game environments offer standardized benchmarks, they either require significant computational resources or lack the complexity necessary to meaningfully engage students in RL experimentation.

To address these gaps, AI Squared aims to create a scalable AI system that allows users to train RL agents within a Brawlhalla-inspired environment while supporting a structured tournament format. The two primary challenges we aim to solve are:

1. **Developing an efficient RL framework that enables users to train and deploy agents in a fighting game environment**: ensuring accessibility, reasonable computational requirements, and clear learning objectives.

2. **Designing a tournament system that allows for competition, collaboration, and iterative learning**: providing users with a platform to improve their agents by testing strategies against peers in a structured format.

By addressing these challenges, AI Squared will bridge the gap between theoretical RL education and practical, hands-on experimentation, fostering engagement and skill development through competitive play.

II. Methodology

Building the Fighting Game Environment - To create the AI Squared RL framework, a platform-fighting game environment inspired by Brawlhalla was developed. The game environment consisted of a platform-fighting game environment (similar to Brawlhalla) and two RL agents on the battlefield that aim to deal damage and knock each other off the platform. The gameplay mechanics is inspired by popular platform-fighting games such as Super Smash Bros, where agents have three lives, percentage/health bars, and different states, such as an agent in a Hurt State (when the agent is attacked) or in a InAirState (when the agent is in the air, and can also jump) which are handled by Finite State Machines (FSM) attached to each agent, controlling which state they are in and which states they may transfer to. During battle, the knockback power delivered by attacks increases proportionally with the damage already taken. The game is over when an agent successfully takes all three lives of the other agent. The gameplay mechanics also introduce the power-cast system, which is a system that controls the state of an attack. For example, for a punching move, the initial power might contain throwing the punch, but it can move into a tree of two other options based on what happens – if it hits, then it will deal damage which causes the other agent to enter a Hurt State, but if it does not hit then it will place the agent in a state of vulnerability where the agent has to be placed on a punch cooldown and can be punished by the opponent. This attack system allows for interesting interactions between the agents to explore. The game also uses a physics-based system to handle realistic interactions between agents and the environment. The observation space of an agent contains information on the agent's (x, y) position, (x, y) velocity, which way the agent is facing, the current state (from FSM), and other life and attack observations that the agent has. Using this information, agents are able to follow the set game rules and mechanics defined by our environment, and players can design intuitive reward functions that are able to maximize the performance of their self-made RL agents. For example, a reward function can be designed using the positions of each agent to reward/punish the agent based on its distance from its opponent. The action space of an agent consists of movement controls and the attacks that the agent can do, and it is tied to a key. Each action is set with data which stores information such as the number of frames for that action or the base damage dealt for attacks. The action space also includes taunting animations, allowing for a fun experience for players.

RL Training Framework - The framework uses the Stable Baselines 3 library, and turns this library into a large custom wrapper to allow for self-play MARL. As such, this framework is able to support agents that use common RL

algorithms supported in the Stable Baselines 3 library, such as PPO, A2C, or RecurrentPPO. Additionally, the framework supports custom-built PyTorch neural network architectures for Deep RL, where users can define simple models such as a basic multilayer perceptron network (MLP) or even advanced architectures such as Transformer models. The framework is also able to support hard-coded agents to interact with the environment. Users who choose to do this can simply write their hard-coded script in the prediction function, and define agent behaviour in the environment by simply extracting observations and performing actions using Helper classes in the framework. The RL framework design promotes a mix of academics and competitiveness, providing users the freedom to approach the tournament how they want. The RL framework is able to support multiple reward functions compiled into a RewardManager class. Users can define multiple reward functions as RewTerm, which takes in a reward function and a weight, which scales the reward based on how important the reward function is to the agent's performance . The user can then compile a dictionary of RewTerms and place it into the RewardManager class, which will return the total accumulated reward given by all reward functions for RL training.

Submission Workflow - Participants in the AI Squared tournament interact with a web interface to submit their agents in the form of an IPython notebook written in Python. Upon submission, the notebook is received by the Tournament Server, where a validation process is initiated. This process simulates the execution of the agent's code within a containerized environment, ensuring the agent functions correctly and monitoring its resource usage, such as CPU and RAM consumption. The Tournament Server runs on a DigitalOcean droplet with 16GB of VRAM and is designed to leverage parallelism while maintaining strict resource constraints for each agent. If any issues are detected during validation, participants are notified and given the opportunity to modify their submissions before resubmission. Once a submission is successfully validated, participants can challenge other users and engage in the double-elimination tournament. The validated submission files are stored in an Azure Blob database for future retrieval.

Challenge Workflow - The challenge mechanism within the tournament follows a pairing system similar to that used by chess.com. Participants may challenge others whose submissions have been validated. If both agents are ready, their code is retrieved from the Azure Blob storage and mounted into the Docker container. The game environment is initialized, and the match runs to completion. At the end of each match, the ELO ratings of the participants are updated to reflect the match outcome. The ELO system allows for dynamic ranking of agents throughout the tournament, enabling continuous performance evaluation. All submission data and results are securely stored in Azure Blob storage, ensuring easy access and management of agent information.

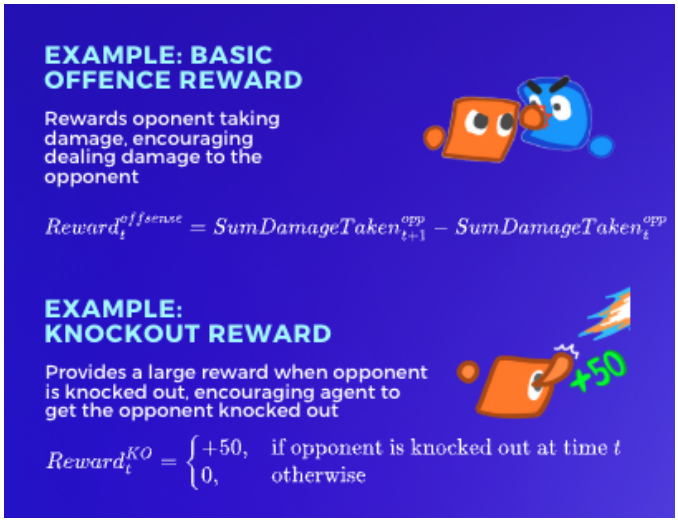

Fig. 2: Rewards implemented in the starter-code notebook, acting as a starting place for reward design. Description of behaviors and graphics included within notebook to help users understand relationship between reward design and behavior.

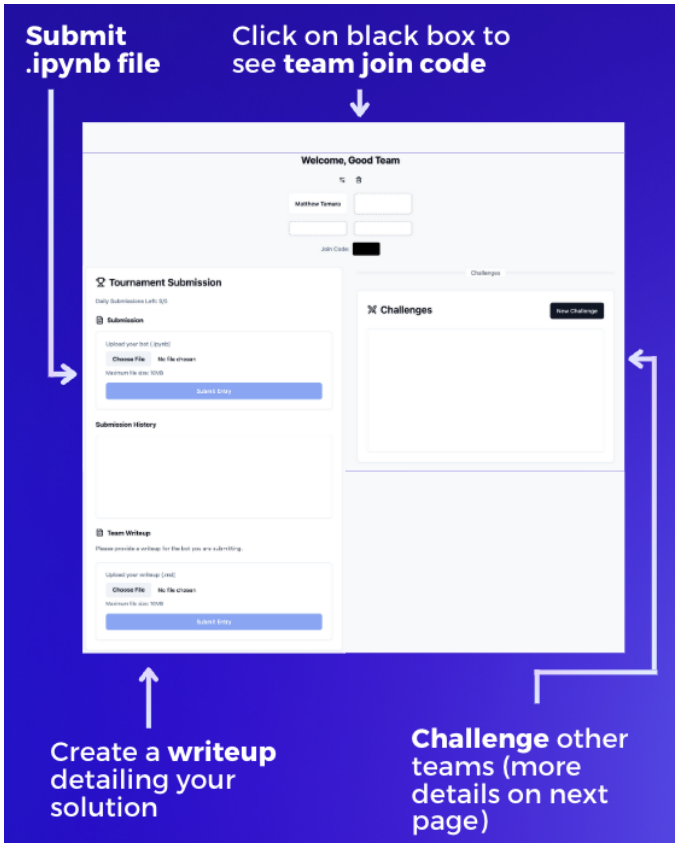

Fig. 3: AI Agent Submission Website: connects to the Tournament Server running on a DigitalOcean droplet

III. Results

Algorithms used in Tournament Submissions - A number of algorithms were used in agent submissions, showcasing a

range of approaches beyond the initial code. These included:

- **Rule-enhanced PPO**: A hybrid approach combining PPO with imitation learning and rule-based actions. This integration enhanced the decision-making of the agent by incorporating predefined behaviors to support performance.
- **Recurrent NEAT (R-NEAT)**: A neuroevolutionary algorithm combined with recurrent neural networks, enabling the agent to adapt and evolve its neural network structure, thereby improving performance in sequential tasks.
- **Population-Based Training (PBT) with Evolutionary Model Merging**: This approach combined evolutionary algorithms with PBT to dynamically adjust hyperparameters and model architectures.
- **Rule-based Algorithms**: These were implemented as simpler agents based on predefined rules for basic actions, serving as a comparative baseline and emphasizing the diversity of algorithmic approaches explored.

A number of these algorithms were not initially implemented in the starter code file (containing a PPO implementation) were created, indicating that people learned and explored a variety of RL algorithms to making agents.

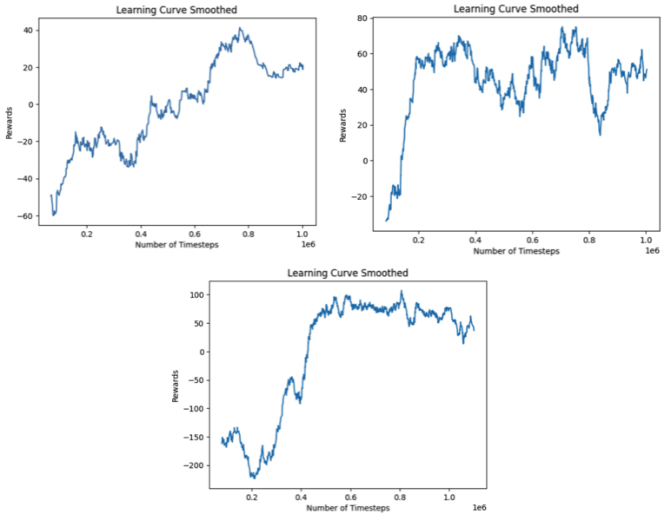

Fig. 4: Learning curves of a Hybrid Reward Function PPO agent with varying reward weightings. The top left emphasizes mid-stage positioning, the bottom reduces penalties for being off-stage, and the top right balances default damage and knockout rewards.

Reward Functions in Tournament Submissions - Using the hybrid reward function framework, participants created and implemented custom reward functions beyond the starter code. This allowed agents to develop new behaviors that were not directly part of the basic design. The reward functions included:

- **Recovery rewards**: Incentivizing agents to successfully recover from knockback or off-stage situations, promoting better survival strategies
- **Combo Rewards**: Rewarding agents for executing complex combos or multi-hit sequences, determined by rewarding longer times of enemy being stunned
- **Desired Positioning/Zoning Rewards**: Encouraging agents to maintain advantageous positions on the map, emphasizing strategic movement and effective control of space.

IV. Conclusion

The AI Squared RL framework enables users to create, train, and battle RL/AI agents. The environment allowed users to define custom reward functions, design RL agents using algorithms like PPO and A2C, and interact with the game through a physics-based system. The system is built around tournament format where participants can challenge others, with their agents being evaluated and ranked using an ELO rating system. Using this framework, users were able to build RL agents capable of performing in a dynamic, competitive environment. The AI Squared RL framework allowed participants to explore a variety of reward functions and RL training techniques in an engaging and educational way.

References

[1] University of Toronto, “ECE411H1 — Adaptive Control and Reinforcement Learning,” Faculty of Applied Science and Engineering, https://engineering.calendar.utoronto.ca/course/ece411h1 (accessed Mar. 17, 2025).

[2] T. M. Moerland, M. Müller-Brockhausen, Z. Yang, A. Bernatavicius, K. Ponse, T. Kouwenhoven, A. Sauter, M. van der Meer, B. Renting, and A. Plaat, ”EduGym: An environment and notebook suite for reinforcement learning education,” *arXiv*, 2024, doi: 10.48550/arXiv.2311.10590. https://arxiv.org/abs/2311.10590.

[3] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, ”OpenAI Gym,” *arXiv*, vol. 1606.01540, Jun. 2016, doi: 10.48550/arXiv.1606.01540.

[4] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaskowski, “Vizdoom: A doom-based AI research platform for Visual Reinforcement Learning,” 2016 IEEE Conference on Computational Intelligence and Games (CIG), pp. 1–8, Sep. 2016. doi:10.1109/cig.2016.7860433

[5] OpenAI, “Gym retro,” Gym Retro, https://openai.com/index/gym-retro (accessed Mar. 17, 2025).

[6] Z. Zhang, S. Akai-Nettey, A. Addo, C. Rogers, and J. Sinapov, ”An augmented reality platform for introducing reinforcement learning to K-12 students with robots,” *arXiv*, Oct. 2021, doi: 10.48550/arXiv.2110.04697.