Duo Ball

Sanjiv Shrirang Joshi, Trym Kyvaag and **Eason Liu**

Khoury College of Computer Sciences

Northeastern University

Boston MA USA

Abstract

This project explores the development and training of reinforcement learning agents to play a simplified 2v2 soccer game using a custom-built environment inspired by the Google Research Football framework. We analyze the possibility of different RL algorithms, notably PPO and DQN, and select PPO based on its relevance. To guide agent behavior, we designed a reward function that goes beyond basic scoring incentives by incorporating heuristics for ball proximity, strategic positioning, and opponent interaction. We further enhanced performance through extensive hyperparameter tuning and iterative evaluation using vector-normalized rewards. We also experimented with curriculum learning, though it showed limited benefit due to the already adversarial nature of our baseline environment. Our final trained PPO agent demonstrates consistent performance after 1M timesteps of training, serving as a foundation for further research.

1 Introduction

Reinforcement Learning (RL) has shown remarkable potential in domains that require agents to learn complex sequences of actions from interaction, such as robotics, autonomous driving, and strategic games. Among the most compelling environments for testing and developing RL algorithms are simulated sports games, which combine elements of real-time decision-making, multi-agent cooperation and competition, spatial awareness, and sparse, delayed rewards.

In this project, we explore the problem of training agents to play a simplified 2v2 soccer game using deep reinforcement learning. Our goal is to develop and evaluate a training pipeline that allows agents (the blue team) to learn to play soccer effectively against an opposing team (the red team) using a custom Gym-compatible environment built with PyGame.

We were motivated by the structure and research potential of environments like the Google Research Football (GRF) Environment (Kurach et al., 2020), which emphasizes realism, customizability, and scalability in RL-based football simulations. However, instead

of leveraging GRF directly, we opted to build our own simplified environment to allow complete control over game dynamics, reward functions, and training loops, enabling faster iteration and better interpretability for our experiments.

The objectives of the project were:

- Design a physics-inspired 2v2 soccer simulation environment suitable for RL experiments depicted in Figure 1.
- Analyze different RL algorithms to identify the most suitable one for our task.
- Engineer and evaluate effective reward functions that align with desirable in-game behavior.
- Explore curriculum learning and hyperparameter tuning strategies to improve sample efficiency and performance.
- Train and evaluate RL agents using Stable-Baselines3 and PPO, aiming for robust and consistent in-game performance.

Through iterative development, reward shaping, hyperparameter tuning, and curriculum learning experiments, we aimed to produce a trained agent that can play the game reasonably well and serve as a solid baseline for further RL research in custom sports environments.

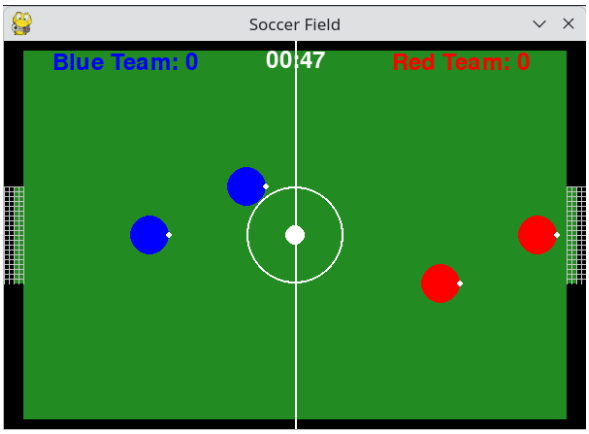


Figure 1: The field with 2 v 2 players and a ball

2 Background/Related Work

Reinforcement Learning (RL) has demonstrated impressive capabilities across a variety of domains by enabling agents to learn from trial-and-error interactions with an environment. In the context of sports simulation and game AI, RL has been instrumental in advancing research in strategy formulation, opponent modeling, and cooperative multi-agent behavior.

A major inspiration for our work is the Google Research Football Environment (GRF) (Kurach et al., 2020), which offers a sophisticated, physics-based 3D football simulator for RL research. GRF highlights key challenges in the domain, such as sparse rewards, long-term planning, and multi-agent coordination. The authors evaluate multiple baseline algorithms—PPO, IMPALA, and Ape-X DQN. Since PPO is broader used algorithm as mentioned in (Kurach et al., 2020) and that this

aligns with broader findings in the RL community that Proximal Policy Optimization (PPO) tends to offer better sample efficiency and stability in continuous control environments compared to Deep Q-Networks (DQN) (described in the footer of this paper on page 1) (Schulman et al., 2017), we decided to choose the PPO algorithm.

While environments like GRF offer complex scenarios and large-scale game simulation, our project focuses on a simplified 2v2 soccer simulation designed for fast experimentation and clearer interpretability. Our implementation is Gym-compatible and inspired by OpenAI Gym-style environments (Brockman et al., 2016), but provides full control over dynamics, rendering, and reward shaping.

Reward shaping has been a critical focus in soccer-based RL. Sparse reward signals, such as scoring a goal, often make learning inefficient. GRF addressed this by introducing the *Checkpoint* reward scheme, which rewards progressive ball movement towards the opponent's goal. In our project, we similarly designed a custom heuristic-based reward function that includes goal scoring, ball proximity, strategic player spacing, defensive behavior, and ball movement direction.

Additionally, curriculum learning has been studied as a way to improve training efficiency by gradually increasing task complexity. For example, Matiisen et al. (Matiisen et al., 2020) proposed a teacher-student framework where simpler subtasks are introduced first to ease learning in complex environments. Inspired by this, we explored curriculum learning by training agents initially in simplified game states (e.g., no opponents or inactive opponents) before introducing more active adversaries. While promising in theory, our implementation of curriculum learning yielded limited gains in practice, as discussed later in the Experiments section.

3 Data

Unlike traditional machine learning projects that rely on static datasets, reinforcement learning agents generate their own training data through interactions with the environment. In our case, the environment is a custom-built 2v2 soccer simulation implemented using PyGame and wrapped as a Gym-compatible interface for integration with Stable-Baselines3.

Observation Space

The environment provides a continuous 15-dimensional observation vector at each timestep, structured as follows:

- **Player positions (x, y):** Each of the

four players (two blue and two red) contributes two values—x and y coordinates—resulting in 8 values in total.

- **Ball position (x, y):** The current x and y coordinates of the ball (2 values).
- **Ball velocity (vx, vy):** The x and y components of the ball's velocity vector (2 values), representing direction and speed of movement.
- **Score (red, blue):** The current game score for both teams (2 values), which are important for shaping strategy in long episodes.
- **Remaining time:** A single value representing the remaining time in the match (in seconds).

Together, this structure provides the agent with full state visibility needed for spatial reasoning, ball tracking, and tactical decision-making.

Action Space

The action space is discrete and multi-agent in nature. Each of the four players (two per team) is controlled independently and can choose from five possible actions:

- **0 – Move forward (up)**
- **1 – Move backward (down)**
- **2 – Move left**
- **3 – Move right**
- **4 – No movement (idle)**

This results in a `MultiDiscrete` action space with shape `[5, 5, 5, 5]`, where each index corresponds to the action taken by one of the four players in a given timestep. In our experiments, only the blue team is controlled by the agent, while red team behavior is limited to interception and pushing ball away from their own goal.

Data Generation and Preprocessing

Training data is generated entirely from the agent's interaction with the environment. Each interaction yields a transition tuple:

$$(s_t, a_t, r_t, s_{t+1}, \text{done})$$

where s_t is the current state, a_t the action taken, r_t the reward received, s_{t+1} the next state, and done indicates whether the episode has terminated.

Preprocessing steps include:

- **Normalization:** Observations and rewards are vector-normalized using Stable-Baselines3's `VecNormalize`, which

helps prevent reward explosion and stabilizes learning.

- **Randomization:** Player starting positions are randomized at the beginning of each episode to encourage generalization and prevent overfitting to specific configurations.
- **Logging:** Custom callbacks log training metrics—such as episode reward, length, and explained variance—into a CSV file for post-training analysis.

Since the environment is fully synthetic and under our control, we have the advantage of unlimited and customizable data generation. This supports rapid prototyping and reproducible experimentation.

4 Methods

To train agents for our 2v2 soccer environment, we built a custom Gym-compatible environment, engineered reward functions aligned with tactical gameplay, and selected Proximal Policy Optimization (PPO) for its stability and sample efficiency in continuous control tasks. All training was done using Stable-Baselines3.

Environment Design

The environment was implemented in PyGame and wrapped to follow the OpenAI Gym API. It supports both visual and headless rendering modes and includes elements such as physics-driven ball dynamics, collision detection, goal scoring, and episode timing.

The action space is defined as `MultiDiscrete([5, 5, 5, 5])`, representing five possible actions (up, down, left, right, idle) for each of the four players. In training, we control only the blue team. Observations are 15-dimensional vectors covering player positions, ball state, score, and remaining time.

Reward Function

We explored both simple and heuristic-based reward functions:

- The initial reward was limited to goal scoring and ball proximity.
- The final heuristic reward incorporated goal outcomes, player-ball distances, team spacing, defensive positioning, and directional ball movement (see `heuristic.py`).

Model Configuration

We used Stable-Baselines3's PPO implementation with a custom multi-layer perceptron (MLP) policy. Key architectural and training parameters included:

- Separate policy and value networks with 2–3 hidden layers
- Initial learning rate of 3×10^{-5}
- Discount factor γ and GAE λ values tuned during training
- Reward normalization and observation scaling via `VecNormalize`

To track progress, we implemented custom reward logging and periodic evaluation callbacks, with checkpointing at regular intervals.

5 Experiments

Our training process involved multiple iterative phases: algorithm selection, reward engineering, curriculum learning, and hyperparameter tuning. Each step was guided by intermediate evaluation and qualitative observations.

5.1 Choosing PPO over DQN

We began by comparing DQN and PPO. DQN was found to struggle in similar environments due to its discrete action value formulation and sample inefficiency in multi-agent, continuous state settings. PPO, by contrast, provided more stable policy updates and better handled the sparse reward structure. This aligns with findings from the GRF environment, which also favored PPO in soccer scenarios (Kurach et al., 2020).

5.2 Reward Function Evolution

The initial reward design used only goal scoring and a distance-to-ball metric. Although agents learned to chase the ball, they failed to develop coordinated or strategic behavior.

The switch to a heuristic reward function (described before) led to noticeable improvements: agents displayed better spatial awareness, moved the ball forward more consistently, and defended their goal area more reliably.

5.3 Hyperparameter Tuning

To further optimize learning, we employed Optuna for hyperparameter tuning. Key parameters included learning rate, clipping range, γ, λ, entropy coefficient, and value loss coefficient.

Tuning especially improved the stability of the value function, as measured by explained variance. We noticed substantial gains after adjusting λ and γ, reducing policy variance and improving reward predictability.

5.4 Curriculum Learning Attempts

We explored curriculum learning by gradually introducing red team opponents—starting with none and scaling up to two active agents. However, the staged approach showed limited benefit. The baseline environment already included sufficient adversarial complexity for effective learning, making curriculum steps redundant.

5.5 Final Training Regime

We trained the final agent for 1 million timesteps. Around 600k steps, reward curves began to plateau, but evaluation metrics such as explained variance and normalized reward remained stable. The final checkpoint was selected at 1M steps based on performance consistency and generalization across random initializations.

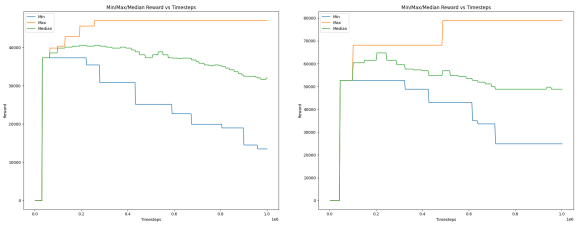
5.6 Reward Curve Comparison: Optuna vs Non-Optuna

To better understand the impact of hyperparameter tuning, we visualized training reward metrics for both the base PPO setup and the Optuna-tuned PPO configuration.

Figure 2 presents two side-by-side subplots:

- The left plot shows mean, median, and max episode rewards over time for the untuned PPO setup.
- The right plot shows the same metrics for the Optuna-tuned model.
- *Important note*, that these values were unnormalized.

This comparison highlights how tuning led to smoother learning, improved peak rewards, and more stable median reward trajectories across episodes.



(a) Non-Optuna PPO (b) Optuna-tuned PPO

Figure 2: Training reward metrics over time: Comparison between Non-Optuna PPO (left) and Optuna-tuned PPO (right). Each plot shows mean, median, and max episode rewards.

6 Results

We evaluated our agent periodically during training using both raw episode rewards and metrics derived from `reward_stats.csv`. The following observations summarize key performance trends:

- **Improved reward trajectory:** The introduction of the heuristic reward and PPO tuning led to substantial increases in average episode reward and more consistent gameplay.
- **Stabilized value learning:** Adjusting γ, λ, and the value loss coefficient significantly improved explained variance, leading to more accurate value estimates and stable policy gradients.
- **Generalization:** The final model maintained strong performance across random initial states and red team behaviors, reflecting successful generalization and robustness.

- **Visual behavior:** Replay evaluations showed coordinated movement, goal-oriented ball control, spacing between players, and fallback defense, which emerged only after reward refinement and tuning.

Our final agent, selected at 1M timesteps, balances reward consistency, explained variance, and observable strategic behavior in evaluation. This checkpoint represents the best trade-off among all our tested configurations.

Figure 3: Training reward per episode over time.

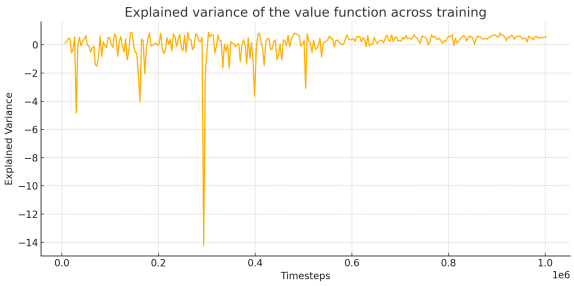


Figure 4: Explained variance of the value function across training.

Our final model, trained for 1 million timesteps, was selected based on stable performance metrics and visual confirmation of strategic gameplay. The model was saved and evaluated over 5 rollout episodes using the replay script.

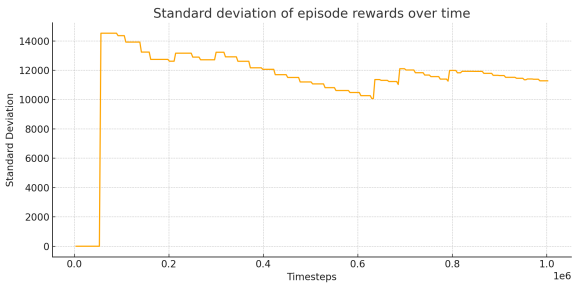


Figure 5: Standard deviation of episode rewards over time.

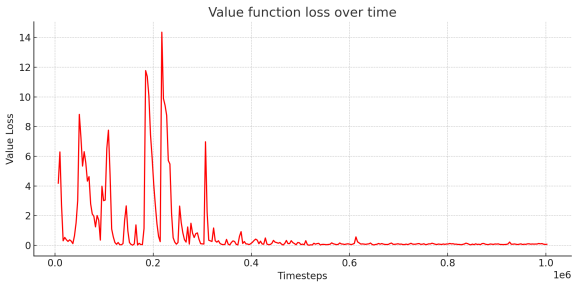


Figure 6: Value function loss across training timesteps. A decreasing trend indicates improved critic estimation, while sharp fluctuations may suggest instability.

6.1 Replay Evaluation Summary

To qualitatively assess the final trained agent, we used the `replay.py` script to simulate two sets of 5 evaluation episodes each. During each episode, the agent played as the blue team against a red team baseline.

The results were as follows:

- **Run 1:** 3 out of 5 wins for the blue team
  - **Episode 1:** 3 goals scored
  - **Episode 3:** 2 goals scored
  - **Episode 4:** 1 goal scored
- **Run 2:** 4 out of 5 wins for the blue team

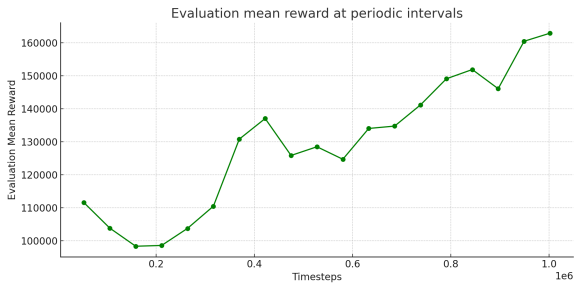


Figure 7: Evaluation mean reward over time, showing the agent's performance in a validation-like environment. This reflects generalization and robustness.

- **Episode 1:** 3 goals scored
- **Episode 2:** 1 goal scored
- **Episode 3:** 1 goal scored
- **Episode 4:** 2 goals scored

These results indicate that the final agent demonstrates consistent offensive behavior and the ability to score multiple goals across games. The replayed episodes further confirmed that the blue team exhibits coordinated positioning and ball pursuit strategies developed during training.

7 Conclusions

In this project, we developed a custom 2v2 soccer simulation environment and trained reinforcement learning agents using Proximal Policy Optimization (PPO). Our work involved building a Gym-compatible multi-agent environment from scratch, engineering effective reward functions, and systematically improving performance through hyperparameter tuning and iterative experimentation.

We observed that simple reward schemes were insufficient for learning strategic behavior. A richer heuristic-based reward function significantly improved coordination, ball progression, and spatial awareness among agents. Curriculum learning, while conceptually appealing, did not yield measurable improvements in our case, likely due to the already challenging baseline environment. Instead, tuning core PPO parameters such as the learning rate, discount factor, and GAE lambda proved to be far more impactful.

Our final agent, trained over 1 million timesteps, demonstrates promising performance in a controlled, adversarial setting. The combination of dense state observations, shaped rewards, and normalized training dynamics enabled robust learning, with evaluation results reflecting stable behavior and policy consistency.

This work provides a foundation for further research into multi-agent RL in simplified sports environments. Future directions could explore opponent modeling, team communication strategies, self-play, or transfer learning from simpler soccer tasks to more complex, coordinated play. Additionally, replacing heuristic rewards with learned reward models or leveraging imitation learning could reduce

the need for manual tuning and promote more generalizable behaviors.

Strengths and Limitations. One of the key strengths of this project is the complete control we had over the environment and training pipeline, allowing for rapid experimentation and detailed customization. Our environment enabled direct observation of agent behavior and fine-grained reward shaping, which would be difficult to achieve in prebuilt environments. However, the project also had limitations. The red team was not co-trained, which may have constrained the level of competitive play. Additionally, the absence of a learned or adaptive reward signal meant that reward shaping remained a manual and time-intensive process. Lastly, due to time constraints, we focused more on tuning than on broader generalization or scalability tests. Addressing these areas could significantly enhance the robustness and realism of future iterations.

References

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. In *arXiv preprint arXiv:1606.01540*.

Karol Kurach, Anton Raichuk, Piotr Stanczyk, Maciej Zajac, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Lars Buesing, et al. 2020. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4501–4508.

Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. 2020. Teacher-student curriculum learning. In *IEEE Transactions on Neural Networks and Learning Systems*, volume 31, pages 3732–3740. IEEE.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. In *arXiv preprint arXiv:1707.06347*.