Multi-Agent Distributed Reinforcement Learning for Playing Soccer with Real Robots

A Project Report
Presented to
The Faculty of the College of
Engineering

San Jose State University
In Partial Fulfillment
Of the Requirements for the Degree
Master of Science in Artificial Intelligence

By
Uttej Kumar Reddy Gade, Prabhath Reddy Gujavarthy, Anjali Sreeja Kadiyala,
Kishore Kumaar Natarajan

December/2023

APPROVED

DocuSigned by:

Stas Tiomkin

5D16C4B2029B444...

Dr. Stas Tiomkin, Project Advisor

ABSTRACT

Multi-Agent Distributed Reinforcement Learning for Playing Soccer with Real Robots
By Uttej Kumar Reddy Gade, Prabhath Reddy Gujavarthy, Anjali Sreeja Kadiyala, Kishore Kumaar Natarajan

This paper introduces an innovative methodology for training Sphero Bolt robots to play soccer. Our approach leverages reinforcement learning algorithms, including deep deterministic policy gradient, multi-agent deep deterministic policy gradient, and proximal policy optimization, with the input data being the robots' positional coordinates. The models undergo training in both single-agent and multi-agent scenarios within a simulated environment using MuJoCo software. Subsequently, successful models are transferred to real Sphero Bolt robots operating in a physical environment. Trained to play soccer in a designated playground, these robots exhibit a commendable success rate in scoring goals while navigating obstacles, relying solely on positional coordinates captured by a camera. This project is a collaborative effort under the CyPhAI group, emphasizing the synergy between various modules: edge devices, cloud infrastructure, virtual replica development, and vision-based tracking. Overcoming challenges such as latency and integration, our findings contribute to the advancement of Sphero Bolts in real-world applications, particularly in robotic competitions and educational settings.

Acknowledgments

The authors are deeply indebted to Professor Stas Tiomkin for his invaluable comments and assistance in the preparation of this study.

Multi-Agent Distributed Reinforcement Learning for Playing Soccer with Real Robots

Uttej Kumar Reddy Gade, Prabhath Reddy Gujavarthy, Anjali Sreeja Kadiyala, Kishore Kumaar Natarajan
Computer Engineering Department
San José State University (SJSU)
San José, CA, USA
Email: {anjalisreeja.kadiyala, kishorekumaar.natarajan, prabhathreddy.gujavarthy, uttejkumarreddy.gade}@sjsu.edu

***Abstract*—This paper introduces an innovative methodology for training Sphero Bolt robots to play soccer. Our approach leverages reinforcement learning algorithms, including deep deterministic policy gradient, multi-agent deep deterministic policy gradient, and proximal policy optimization, with the input data being the robots' positional coordinates. The models undergo training in both single-agent and multi-agent scenarios within a simulated environment using MuJoCo software. Subsequently, successful models are transferred to real Sphero Bolt robots operating in a physical environment. Trained to play soccer in a designated playground, these robots exhibit a commendable success rate in scoring goals while navigating obstacles, relying solely on positional coordinates captured by a camera. This project is a collaborative effort under the CyPhAI group, emphasizing the synergy between various modules: edge devices, cloud infrastructure, virtual replica development, and vision-based tracking. Overcoming challenges such as latency and integration, our findings contribute to the advancement of Sphero Bolts in real-world applications, particularly in robotic competitions and educational settings.**

***Index Terms*—Sphero Bolt, Robotics, MuJoCo, Multi-agent reinforcement learning.**

I. Motivation

Reinforcement Learning, a subsection of artificial intelligence, has been on the rise in terms of research due to advancements in deep learning. In this machine learning paradigm, an agent learns how to make decisions by interacting with an environment, with its main aim being accumulating maximum reward. Its real-world applications range from robotics and autonomous vehicles to finance and healthcare. This could also be used to train agents to play sports like soccer. Our motivation behind this paper is to enable the utilization of reinforcement learning to train easily available bots like Sphero Bolt to play soccer. This allows experimenting with various reinforcement learning algorithms, and integration between a simulated and physical environment, as well as an opportunity to participate in robotic competitions. Distributed Multi-Agent Reinforcement Learning (MARL) holds significant importance in various domains. This project mainly focuses on Distributed Reinforcement Learning and working on multi-agent systems.

II. Introduction

Playing soccer with robots is a formidable challenge that has long appealed to the public imagination. Popular international competitions like RoboCup [16] [8] have cropped up in response since 1997 to progress research in the field of robotics, AI and multi-agent cooperation. Historically this challenge was solved through handcrafted strategies with actions such as kicking, dribbling and passing taken based on predetermined scenarios. With the rapid progress of machine learning in recent times though, more solutions are utilizing these algorithms to optimize the aforementioned actions in terms of timing [14] [24]. Reinforcement learning, an interdisciplinary area of machine learning, has also proved to be effective in approaching this problem, though not as effective as handcrafted strategies, yet, due to its inherent limitations. However, the area has a lot of promise in that players learn to play soccer without previous knowledge of game-specific strategies leading to more dynamic plays. In this paper, reinforcement learning algorithms are explored in a simulated soccer setting and the learned models are applied to Sphero Bolt robots [20] in a physical environment.

MuJoCo [21], an advanced physics simulator is used to simulate a soccer environment and train the reinforcement learning models. The arena is adapted from the rule book of RoboCup Small Size League [15], for want of standardized dimensions. The players are modeled after Sphero Bolt robots and the soccer ball is modeled after a table tennis ball. In an effort to simplify the simulation, the players are given an action space of size 2 and full observability of all relevant elements in the game. DDPG [9], PPO [18] and MADDPG [13], algorithms for continuous action space are used to train the agents in various scenarios, like moving to a point, avoiding obstacles and scoring a goal.

The trained models are then applied to Sphero Bolts in a 5x5 unit bounded area. A camera positioned above the arena sends images periodically to a server which detects attributes like positions of the robots and ball, constructs the observation space and gets the optimal action from the policy of the trained model. This action is then sent to the Sphero Bolts via a bluetooth connection and the loop continues. This paper provides a MuJoCo soccer environment and demonstrates the application of reinforcement learning algorithms and provides a framework to apply the virtual models to physical sphero bolt robots.

III. Problem Statement

Through this project, we want to enable Sphero Bolt robots to play soccer using reinforcement learning. It aims to develop a virtual soccer environment adhering to the tournament rules and specifications using MuJoCo simulation software. Later trained the single virtual Sphero Bolt agent to first reach a particular point on the simulated playground. Then after addition of obstacles, the agent is trained again, and now is able to avoid them successfully. Finally, the single agent is optimized to move the ball to the goal. This is integrated to the physical environment. Next phase is changing this to multi-agent. Using the weights of the single agent models, 3 Sphero Bolt virtual agents were trained to successfully reach the ball. Next phase is to incorporate MADDPG algorithm for multi-agent training. Finally, integrate the policies learned to the physical Sphero Bolts and adjust the learning based on the observations. We work as part of the CyPhAI group, which involves teams working on other aspects of the project (edge devices, cloud, virtual replica, vision-based tracking and UI) to achieve the integrated soccer environment.

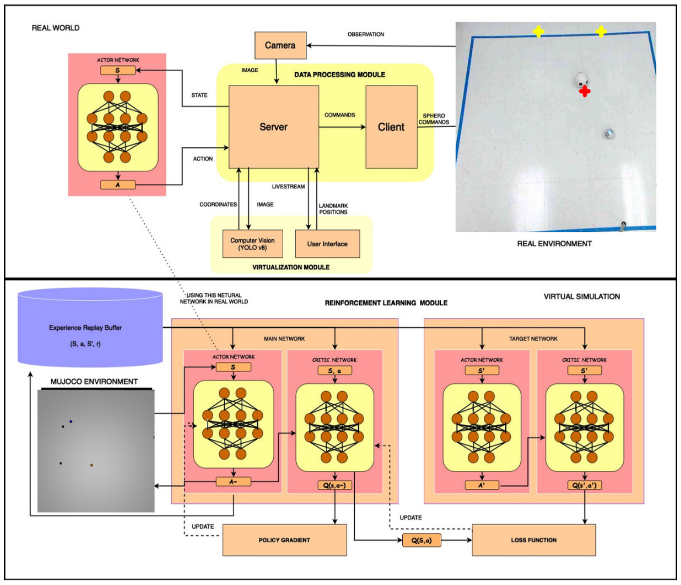
IV. Related Works

The problem of playing soccer with robots gained momentum from the announcement of RoboCup competitions in 1997 [8]. The teams participating in this tournament are required to publish their research and implementation details in the form of Team Description Papers [17]. Of particular interest are the papers of ER-Force [7], a 4-time finalist team who took the route of multi agent systems in their solution.

Till 2013, the team approached the AI of the game with a Skills-Tactics-Plays [2] approach, which is hierarchical architecture designed to control robots in an adversarial environment. However, the approach imposed several problems when scaling which prompted the team to look into multi-agent systems [1] which is what we are interested in too. Further progression in this strategy is detailed in the following year's papers with justifications and implementations of key features like path planning, obstacle handling [5], group training [5], ball mechanics modeling [11] and flight tracking [3] [22], midfield strategies [6] and message passing [12].

Having noted the important components which comprise a successful robot AI system, the research then turned to papers that focused on multi-agent reinforcement learning, especially in the field of cooperative-competitive environments. Multi-Agent Deep Deterministic Policy Gradient, MADDPG showed success in training agents effectively in environments that have shared and conflicting objectives [13]. Multi-Agent Proximal Policy Optimization [23] is another algorithm that proved to be effective in this area.

More recently, Google's DeepMind implemented an end-to-end multi agent reinforcement learning algorithm which demonstrated agents' cooperating with decentralized, population-based training with co-play [10]. The environment was implemented in MuJoCo with the game being a 2 vs 2 semi-humanoid players. For simplying the physics though, contacts between the players was removed. However, as this paper focuses also on applying the learned models to Sphero



begin
center

Fig. 1. Project Architecture

Bolts, the contacts between the players will be on in the simulation. However, this would require extensive computation. To ease this to an extent, the methods as outlined in [19] was explored. These methods involve parallel computation, polyak averaging and policy batching.

To speed up the training and testing of the algorithms, a simulation software. The software should be able to model the sphero bolts and the ball behavior as close to the real-world as possible so that it becomes easier to apply it on sphero bolts later on with minimal change. Between Bullet [4] and MuJoCo [21], two of the best free simulation software, MuJoCo was chosen for its relatively more accurate modeling and the ease of its integration with the rest of the software.

V. Project Architecture

The project architecture is designed for a sophisticated integration between three modules: Reinforcement Learning Module, Virtualization Module and Data Processing Module, to implement real-time interactions between the real-world interactions and the intelligent agent decision making work together to enable robots to play soccer.

A. Reinforcement Learning Module

The Reinforcement Learning (RL) module within this decision-making processes in a simulated environment, eventually translating this learning into real-world execution. Primarily built upon the principles of Deep Deterministic Policy Gradient (DDPG) and Multi-Agent Deep Deterministic Policy Gradient (MADDPG) architectures, this module leverages MuJoCo and OpenAI Gym to create and simulate environments for training.

The core of the RL module revolves around DDPG, an actor-critic framework that involves a continuous action space and a learning approach based on experience replay. In the case of MADDPG, the number of actor networks aligns with the number of agents within the environment, thereby enabling each agent to have its own policy network.

During the interaction procedure, the primary actor-network, representing an agent, engages with the simulated environment. It takes in the current state of the environment and generates an action based on this information. Upon execution of this action, the environment provides immediate feedback in the form of a reward and the subsequent state resulting from the action.

During real-world execution, the actor-network from the main network is employed for decision-making processes, leveraging the learned policies and experiences gathered during the training phase.

B. Virtualization Module

Virtualization Module consists of two sub-components: computer vision and User Interface (UI). Computer vision gets images, and analyzes them using the YOLOv8 model to detect objects, including the ball and Sphero Bolt robots, and determine their coordinates. The User Interface also gets images from the server and sends continuously for the live stream to the User interface and returns the positions of the landmarks (such as goal poles), and target positions.

C. Data Processing Module

The Data Processing Module serves as an intermediary, facilitating seamless data exchange between Real world Environment, Virtualization Module and Actor Network. To achieve this seamless data exchange, the Data Processing Module employs websockets, a bidirectional communication protocol that enables real-time data streaming. Through websockets, the module continuously receives image data captured by the camera, which provides a real-time representation of the surrounding environment.

This real-time image data is then transmitted to the Virtualization Module. In return, the Data Processing Module receives information from the Virtualization Module, including coordinates, landmark positions (such as goal poles), and target positions. The transformation of image-based coordinates into an abstract coordinate system is a critical step in preparing the data for the Reinforcement Learning Module.

This transformation ensures that object positions are represented in a format that aligns with the Reinforcement Learning Module's requirements, facilitating efficient learning and decision-making processes. These decisions of which action to take is sent to the client through which it is communicated to the sphero robots.

The RL framework uses this information to determine the optimal actions using a pretrained model. The RL framework is implemented using the Deep Deterministic Policy Gradient (DDPG) algorithm, complemented by the Multi Agent Deep Deterministic Policy Gradient (MADDPG)approach. The actor-critic approach within MADDPG allows efficient policy improvement in continuous action spaces within a complex environment. The actor-critic approach ensures that both the current and long-term outcomes are considered for an intelligent behavior. The core of the RL framework is the feedback loop where the actions of our agents are continuously evaluated and adjusted based on real-time feedback. The replay buffer implemented in this loop allows the agents to learn from past experiences and improve their actions over time. The interaction between observation, decision-making, action, and evaluation forms the backbone of our system, forming a sophisticated and continuous learning process.

VI. Base Implementation

A. Creating Simulations using MuJoCo

The soccer environment is simulated in MuJoCo to closely replicate real-world physics, using different environments for different scenarios. The environment is designed comprehensively, with appropriate elements like field dimensions, object properties, and other configurations like friction and bounce clearly defined to simulate a realistic soccer ground. The environments are created using the xml file and executed using the python.

B. MuJoCo-Gym Integration

The MuJoco environment was integrated with the OpenAI Gym framework, for training and testing the RL algorithms. This integration provided a consistent and structured interface for the exchange of state and action information.

C. Reward System

The rewards were designed to guide and improve the agent's learning trajectory over time. Starting with the basic rewards for approaching the ball, we implemented complex reward functions for pursuing objectives for obstacle avoiding, and goal scoring. This system was crucial in incrementally shaping the agent's skills and decision making capabilities.

D. Learning Algorithms

1) Deep Deterministic Policy Gradient (DDPG): DDPG is a model-free, off-policy actor-critic algorithm that yields deterministic actions rather than probabilities. This aspect was most critical to ensure precise and decisive movements of the agent which is crucial for tasks like ball handling and goal-scoring. The implementation of DDPG was central to our single-agent methodology, with special attention to the integration of a replay buffer and the establishment of a feedback loop. DDPG is more suited for continuous action spaces, which allowed for more control of the agent's movements. DDPG's dual nature of learning both a policy function(actor), and a value function(critic), helped our algorithm to balance between exploring new behaviours and exploiting known successful behaviours. This was essential for the agent to learn and improve its performance over time in the soccer environment. This implementation of DDPG was central to our single-agent methodology, with special attention for the establishment of Feedback loop.

Feedback Loop — The feedback loop mechanism in DDPG has two components, Actor-Network and Critic-Network,

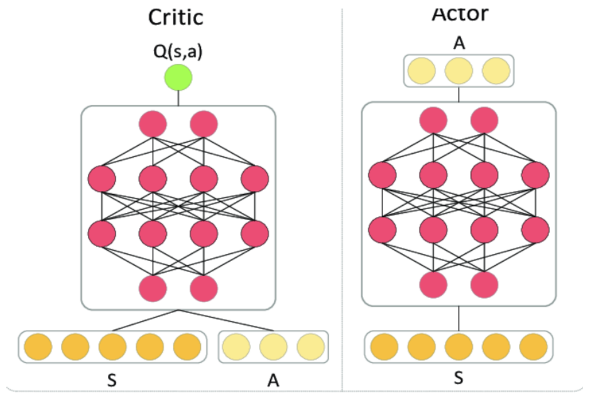


Fig. 2. Deep Deterministic Policy Gradient

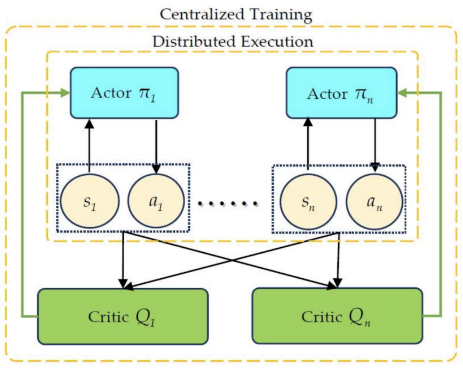


Fig. 3. Multi-Agent Deep Deterministic Policy Gradient

which was essential for iterative learning and policy improvement. This continuous feedback ensures that the agent not only learns but also adopts to new objectives.

Replay Buffer Integration — This buffer stores a history of agent's experiences(current state, action, reward, next state etc.). By randomly sampling from this buffer to train the agent, we ensured stability and efficiency of the learning process.

Hyperparameter Tuning — Parameters like learning rates, discount factors, and exploration rates were carefully calibrated to balance the speed of learning with the stability and effectiveness of the learned behaviors.The carefully balances exploration (trying new actions) and exploitation (using known successful actions).

2) *Multi-Agent Deep Deterministic Policy Gradient (MADDPG):* MADDPG extends the principles of DDPG to multi-agent scenarios, addressing the challenges and dynamics of environments where multiple agents interact and learn simultaneously. This extension is particularly relevant in scenarios like playing soccer, where teamwork, competition, and interaction dynamics are important.

Decentralized Actor with Centralized Critic: MADDPG operates with a structure where each agent has its own actor-network (policy function) but shares a centralized critic network (value function) across agents. MADDPG's framework allows agents to make informed decisions with limited information. The learning process in MADDPG involves each agent improving its policy while considering the potential actions and policies of other agents. The shared critic network maintains stability by providing a consistent learning signal to all agents.

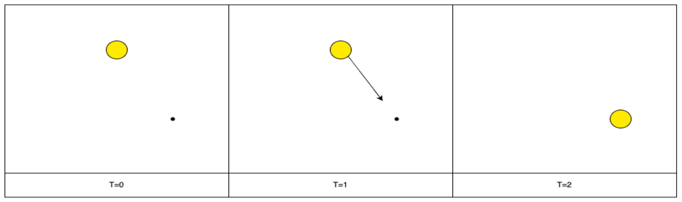


Fig. 4. Single Agent Navigation

E. *Visualization of Agent's behavior and Analysis*

To ensure consistent improvement of the agent's performance, it is crucial to ensure continuous monitoring and analysis of its behaviors. Detailed Learning and Loss curves were plotted at regular intervals, to assess agent's progress in identifying successful strategies and the right combination of hyperparameters.

VII. Single agent Implementation

Single Agent Implementation focuses on individual learning and decision-making. It's simpler but less representative of real-world scenarios where multiple agents must cooperate or compete.

A. *Phase 1: Single Agent Navigation without Obstacle*

Objective: Train the agent to reach a specified final state from a random initial state in a straightforward environment without obstacles.

Approach: The agent is programmed to understand its position within the virtual soccer environment and learn to move towards specific locations.

Reward System: The agent receives positive feedback for reducing the distance to the target and negative feedback for increasing it, guiding it to learn efficient path navigation.

The image shows a yellow circle and a black dot. The yellow circle is the agent and the black dot is the target. The agent is trained to reach a random final state from a random initial state. T=0 refers to timestamp 0.

- **T=0:** The agent starts in a random initial state.
- **T=1:** The agent decides on an action to move to a new state.
- **T=2:** The agent moves to the targeted state.

The agent starts at a random initial state and tries to reach the target state in as few steps as possible. The agent is rewarded for reaching the target state and penalized for taking too many steps. The agent learns to reach the target state by trial and error.

Training Steps:

1) The agent begins in a random initial state.
2) It chooses an action to transition to a new state.
3) The agent receives a reward or penalty based on whether the target state is reached or not.

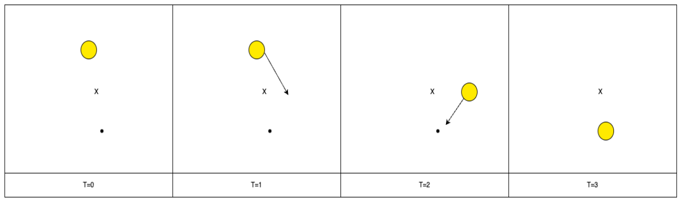


Fig. 5. Single Agent Navigation with obstacle

4) Steps 2 and 3 are repeated until the target state is reached or a maximum step limit is surpassed.
5) The agent's policy is updated according to the rewards and penalties received. The policy is a function mapping states to actions, adjusted to favor actions that led to higher rewards in the past.

After training, the agent can be used to generate trajectories from random initial states to random final states.

B. Phase 2: Single Agent Navigation with Obstacles

Objective: Adapt the agent to reach a specified final state from a random initial state, but with the added complexity of obstacles.

Challenge: The agent needs to learn how to navigate around obstacles while still aiming to reach the target location.

Strategy: The training involves enhancing the agent's ability to perceive and react to environmental changes, ensuring it can adapt its path in real-time to avoid obstacles.

Training Steps:

1) The agent is placed in a random initial state.
2) The agent selects an action based on its policy.
3) The agent takes the action and observes the new state.
4) The agent receives a reward or penalty depending on whether it reached the target state, collided with an obstacle, or took too many steps.
5) The agent's policy is updated based on the reward it received and the state transition.
6) Steps 2 to 5 are repeated until the target state is reached or a maximum step limit is surpassed.
7) The agent's policy is updated according to the rewards and penalties received. The policy is a function mapping states to actions, adjusted to favor actions that led to higher rewards in the past.

The image shows a yellow circle and a black dot. The yellow circle is the agent, cross mark is the obstacle and the black dot is the target.

- **T=0:** The agent starts in a random initial state.
- **T=1:** The agent avoids to obstacle on moves to a new state.
- **T=2:** The agent decides on an action to move to a new state.
- **T=3:** The agent moves to the targeted state.

C. Phase 3: Single Agent Hitting the Ball Towards the Goal

Objective: Train the agent to not only reach a target location but also to hit a ball towards the goal.

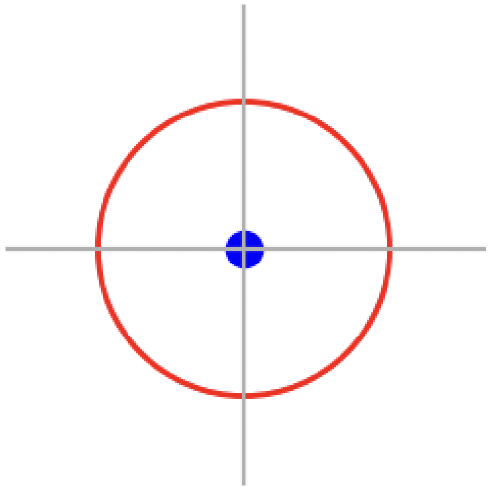


Fig. 6. Proximity Circle

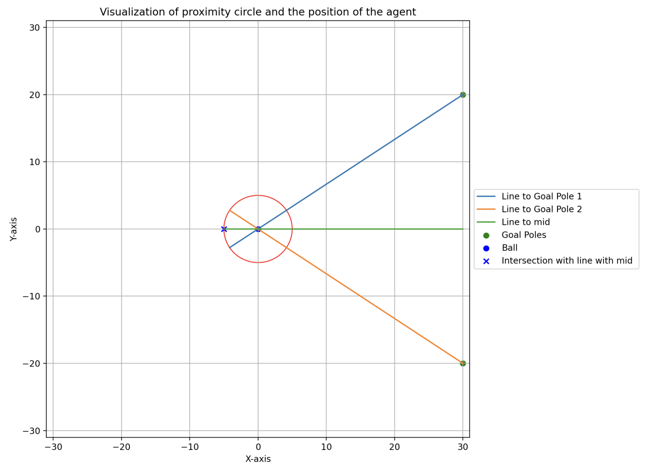


Fig. 7. Visualization of proximity circle and position of the agent

Reward system: Rewards are given for moving closer to the ball and successfully directing it towards the goal, with penalties for unsuccessful attempts or moving away from the target.

Calculating the target based on the ball's position:

The target is calculated with the help of football's position. Considering the football is the center of the proximity circle. The proximity circle represents a circumference within which the agent should touch and remain, to optimize its chances of successfully intercepting the ball at a point that would maximize its chances of hitting the goal.

The line from the middle point of goal poles passes through the center of the circle which intersects the circle in 2 points on the circle we have to find the point which is not in between the circle will be the target point to the agent.

Training Steps:

1) The agent is placed in a random initial state.
2) The agents calculates the target based on the ball's position using the proximity circle.
3) The agent selects an action based on its policy.
4) The agent takes the action and observes the new state.

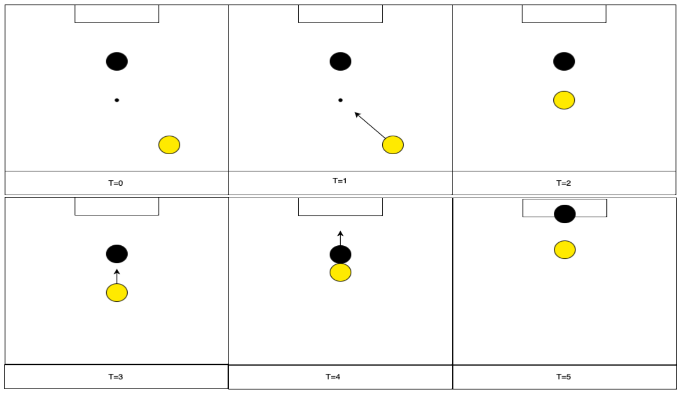


Fig. 8. Single Agent Soccer Game

5) The agent receives a reward or penalty depending on whether it reached the target state, collided with an obstacle, or took too many steps.
6) The agent's policy is updated based on the reward it received and the state transition.
7) Steps 2 to 6 are repeated until the ball reaches the goal state is reached or a maximum step limit is surpassed.
8) The agent's policy is updated according to the rewards and penalties received. The policy is a function mapping states to actions, adjusted to favor actions that led to higher rewards in the past.

The image explains agent hitting the ball to the goal where yellow is the agent, black dot is the football and the black box is goal post.

- **T=0:** The agent starts in a random initial state and calculates the target.
- **T=1:** The agent decides on an action to move to a new state.
- **T=2:** The agent moves to the targeted state.
- **T=3:**The agent kicks the football towards the goal post.
- **T=4:**The football travels towards the goal post.
- **T=5:** The football reaches the goal post.

VIII. Multi agent Implementation

Multi-Agent Implementations involves multiple agents learning simultaneously. This approach is more complex due to the interactions between agents, requiring sophisticated co-ordination and strategy development. It's more representative of real-world team-based tasks.

A. Phase 4: Multi-Agents Navigation

Objective: Train multiple agents to navigate to different locations without interfering with each other.

Complexity: This phase introduces challenges of coordination and spatial awareness among multiple agents.

Collaborative Strategy: The agents must learn to navigate the field in a way that maximizes coverage and efficiency while minimizing collisions and interference with each other's paths.

The image shows a yellow circle, orange circle, blue circle and a black dots. The yellow circle, orange circle, blue circle are the agent and the black dots are the target. The agent is

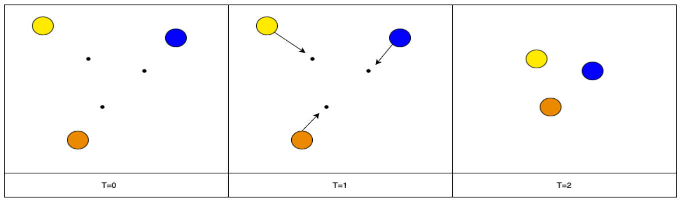


Fig. 9. Multi Agent Navigation

trained to reach a random final state from a random initial state. T=0 refers to timestamp 0.

- **T=0:** The agents starts in a random initial states.
- **T=1:** The agents decides on an action to move to a new states.
- **T=2:** The agents moves to the targeted states.

The agent starts at a random initial state and tries to reach the target state in as few steps as possible. The agent is rewarded for reaching the target state and penalized for taking too many steps. The agent learns to reach the target state by trial and error.

Training Steps:

1) The agents begins in a random initial states.
2) It chooses an action to transition to a new states.
3) The agents receives a reward or penalty based on whether the target states is reached or not.
4) Steps 2 and 3 are repeated until the target states is reached or a maximum step limit is surpassed.
5) The agents policies are updated according to the rewards and penalties received. The policy is a function mapping states to actions, adjusted to favor actions that led to higher rewards in the past.

After training, the agents can be used to generate trajectories from random initial states to random final states.

B. Phase 5: Multi-Agents Hitting the Ball Towards the Goal

Objective: Train multiple agents to collaboratively hit the ball towards the goal.

Teamwork and Timing: Agents must learn not only to navigate the field but also to coordinate their actions to hit the ball effectively towards the goal.

Advanced Training: This phase involves complex training where agents must understand each other's positions and movements, predict the trajectory of the ball, and execute synchronized actions for successful gameplay.

The state which is the observation space consists of 16 components for each agents (three in our case). They are x axis-velocity of the agent, y axis-velocity of the agent, x-coordinate of the agent, y-coordinate of the agent, relative x-coordinate of the other agents, relative y-coordinate of the other agents, relative x-coordinate of the random points, relative y-coordinate of the random points, relative x-coordinate of the ball, relative y-coordinate of the ball. The action space is 8 actions same as single agent due to computations. The environment is trained using MADDPG (Multi-Agent Deep Deterministic Policy Gradients).

Calculating the targets based on the ball's position:

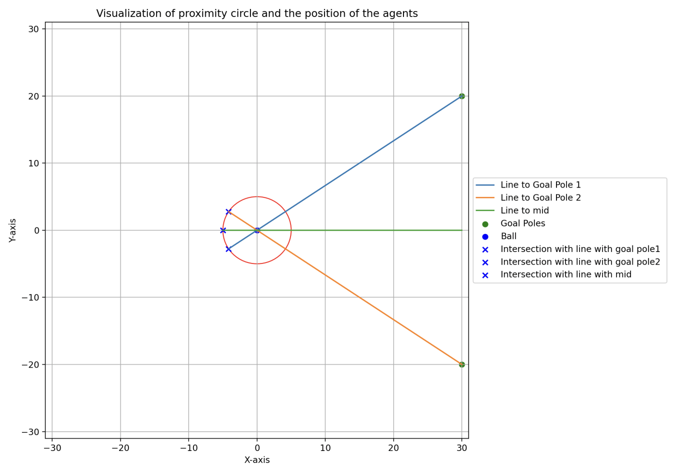


Fig. 10. Visualization of proximity circle and position of the agents

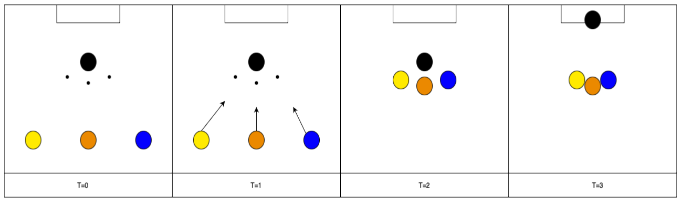


Fig. 11. Multi-Agents Collaborative Soccer Game

The targets are calculated with the help of football's position. Considering the football is the center of the proximity circle. the positions of all the three agents on the proximity circle similar to the case of single agent. The only difference from the single agent will be the algorithm finds the point of intersection of lines passing the ball from both the goal poles. So, Now instead of agents going to the randomly generated points, it uses the points generated through the algorithm as shown in figure.

Training Steps:

1) The agents is placed in a random initial states.
2) The agents calculates the target based on the ball's position using the proximity circle.
3) The agents selects an action based on its policies.
4) The agents takes the action and observes the new state.
5) The agents receives a reward or penalty depending on whether it reached the target state, collided with an obstacles, or took too many steps.
6) The agents policies are updated based on the rewards it received and the state transitions.
7) Steps 2 to 6 are repeated until the ball reaches the goal state is reached or a maximum step limit is surpassed.
8) The agents policies are updated according to the rewards and penalties received.

The image explains agent hitting the ball to the goal where yellow circle, orange circle, blue circle and a black dot. The yellow circle, orange circle, blue circle are the agents, black dot is the football and the black box is goal post.

- **T=0:** The agents starts in a random initial states and calculates the targets.

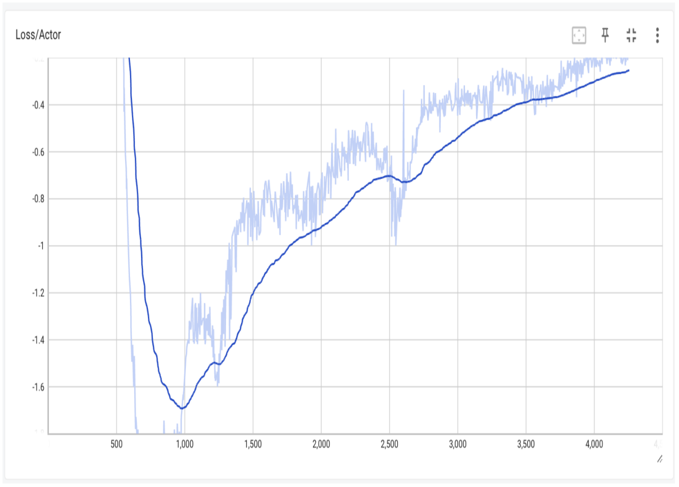


Fig. 12. Actor Loss

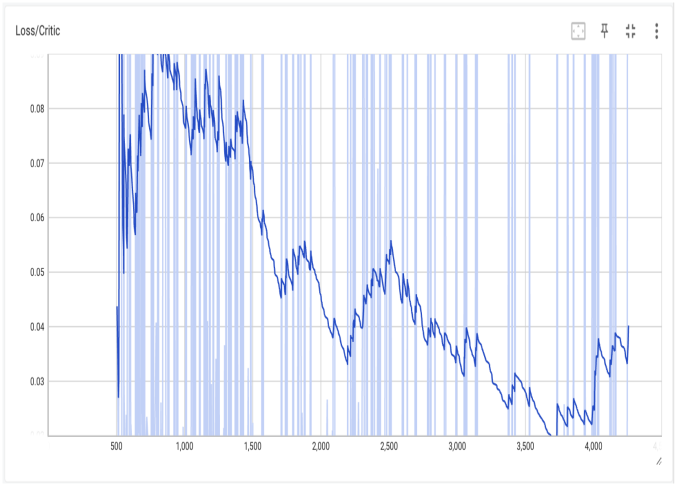


Fig. 13. Critic Loss

- **T=1:** The agents decides on actions to move to a new states.
- **T=2:** The agents moves to the targeted states.
- **T=3:**The agents kicks the football towards the goal post.
- **T=4:**The football reaches the goal post.

IX. Results

In the single-agent scenario, our application of the Deep Deterministic Policy Gradient (DDPG) algorithm yielded promising results. The primary objective was to train an agent to navigate to a specific point within the MuJoCo simulated environment. After an extensive training period of over 72 hours, we achieved a commendable success rate of 94% in reaching the target point.

Building on this success, we further challenged the agent by introducing obstacles along its path. Despite the increased complexity, the agent exhibited robust adaptability, achieving a success rate of approximately 87% in navigating to the designated point while overcoming obstacles. Subsequently, we enhanced the agent's capabilities to include ball movement towards the goal, even in the presence of three additional

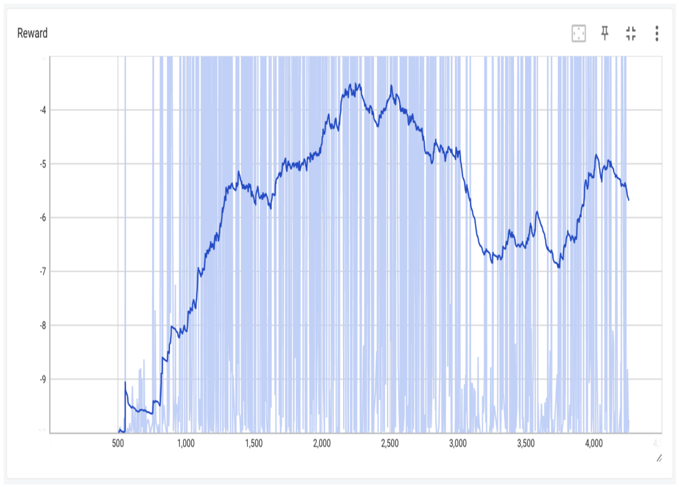


Fig. 14. Reward for single agent with obstacle using DDPG.

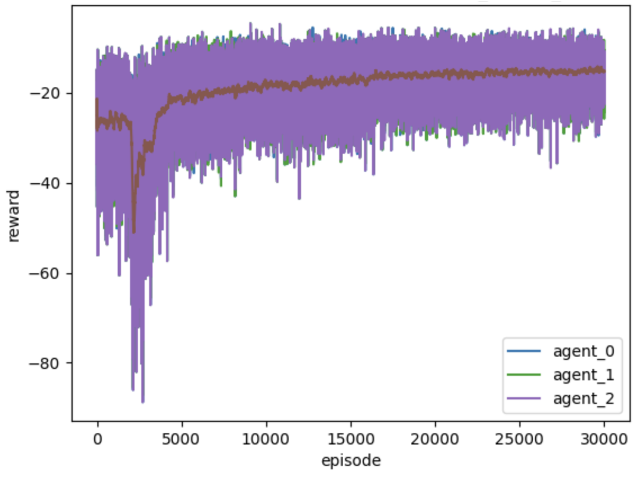


Fig. 15. Episodic Reward for 3 agents using MADDPG.

obstacles. Remarkably, the success rate showed only a modest reduction, showcasing the versatility and robustness of the trained model.

The successful outcomes in the simulated environment paved the way for a seamless transition to the physical environment. The trained agent demonstrated its learned behaviors effectively in the real-world setting, overcoming challenges such as latency, friction, and other dynamics inherent to the physical realm.

To extend our model to a multi-agent setting, we introduced three additional agents into the simulated environment. Leveraging the optimal weights obtained from the best-performing single agent, we conducted training sessions for each of the additional agents. The results were promising, with three agents successfully reaching the position of the ball and actively engaging in collaborative efforts to advance it towards the goal.

Post this, the model has a common buffer, which would enable us to have centralized training. This gave us decent results as well. The reward plot can be seen in 15.

These findings highlight the adaptability and transferability of the single-agent model to a multi-agent context, opening avenues for cooperative behaviors among multiple agents in pursuit of a shared goal.

X. Challenges Encountered

A. *Limited Coordination*

Each of the three agents with its own replay buffer operates in relative isolation, lacking direct access to the experiences and observations of other agents. This isolation can lead to suboptimal coordination among agents.

B. *Incomplete Environment Understanding*

Since each agent maintains its own replay buffer, they only have access to their individual perspectives of the environment. This fragmented view limits their ability to understand the global state of the environment and hampers their capacity to make decisions that consider the actions and intentions of other agents.

C. *Increased Sample Inefficiency*

Maintaining three separate replay buffers may lead to increased sample inefficiency. Agents may need more training samples to learn effective policies due to the limited diversity of experiences in their individual buffers. This inefficiency can slow down the learning process and hinder the overall performance of the multi-agent system.

D. *Scalability Issues*

As the number of agents increases, managing individual replay buffers for each agent becomes more complex and resource-intensive. This scalability challenge can hinder the application of the multi-agent system to larger and more intricate environments.

XI. Conclusion

In conclusion, our exploration into applying reinforcement learning to train Sphero Robots is successful with decent results various experimental scenarios. Our project mainly contributes to the broader landscape of RL research by extending its application domain to more accessible robotic platforms like Sphero Bolt. Through experiments, we have delved into various challenges of coordinating multiple agents, transferring knowledge from simulations to the physical world. Our paper serves as a starting point for the research community, students, and other enthusiasts to further explore the potential of single and multi agent reinforcement learning in accessible robots.

XII. Future Work

The limitations could be avoided by using multi-agent deep deterministic policy gradient (MADDPG) algorithm, where all the agents share a common replay buffer. This ensures that their training is central, despite decentralized execution, thus improving the global understanding. MAADPG also enables more consistent exploration strategy among the agent.

Investigating adaptive learning rate mechanisms within MADDPG may contribute to maintaining consistent learning across agents. Fine-tuning learning rates based on the evolving dynamics of the multi-agent system could mitigate the risk of divergent learning strategies.

Assessing the scalability of MADDPG to larger multi-agent systems could also be an avenue for future research. Understanding how well the algorithm performs as the number of agents increases can provide valuable insights for its applicability in more extensive and intricate environments.

Acknowledgment

The authors would like to thank our advisor, Dr. Stas Tiomkin, and the Charles W Davidson College of Engineering at San Jose State University.

References

[1] H. Bayerlein and et al. “ER-Force Team Description Paper for RoboCup 2014”. In: 2014.

[2] B Browning et al. “STP: Skills, tactics, and plays for multi-robot control in adversarial environments”. In: *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*. 2005, pp. 33–52.

[3] J. Bühlmeyer and et al. “ER-Force Extended Team Description Paper RoboCup 2017”. In: 2017.

[4] E. Coumans. “Bullet Physics Simulation”. In: *SIGGRAPH ’15: ACM SIGGRAPH 2015 Courses*. July 2015, art. 7.

[5] M. Eischer and et al. “ER-Force Extended Team Description Paper RoboCup 2015”. In: 2015.

[6] T. Engelhardt and et al. “ER-Force 2019 Extended Team Description Paper”. In: 2019.

[7] Robotics Erlangen. *Small Size League*. Accessed: November 28, 2023. Year. URL: https://www.robotics-erlangen.de/en/small-size-league-2.

[8] H. Kitano et al. “RoboCup: The Robot World Cup Initiative”. In: *Agents*. 1997, pp. 340–347.

[9] T. P. Lillicrap and et al. “Continuous Control with Deep Reinforcement Learning”. In: *arXiv preprint arXiv:1509.02971* v6 (May 2019).

[10] S. Liu et al. “Emergent Coordination through Competition”. In: *International Conference on Learning Representations*. 2019.

[11] C. Lobmeier and et al. “ER-Force Extended Team Description Paper RoboCup 2016”. In: 2016.

[12] C. Lobmeier et al. “ER-Force 2018 Extended Team Description Paper”. In: 2018.

[13] R. Lowe et al. “Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments”. In: *Advances in Neural Information Processing Systems 30*. 2017.

[14] M. Prokopenko and P. Wang. “Guiding Self-Organisation of Intelligent Agents”. In: *Fractals2019, RoboCup 2019: Robot World Cup XXIII*. 2019.

[15] RoboCup. *RoboCup Small Size League Rules*. Accessed: November 28, 2023. Year. URL: https://ssl.robocup.org/rules/.

[16] RoboCup. *RoboCupSoccer*. Online; accessed November 28, 2023. 2023. URL: https://www.robocup.org/domains/1.

[17] RoboCup. *Team Description Papers*. Accessed: November 28, 2023. Year. URL: https://ssl.robocup.org/team-description-papers/.

[18] J. Schulman and et al. “Proximal Policy Optimization Algorithms”. In: *arXiv preprint arXiv:1707.06347v2* v2 (28 Aug 2017).

[19] A. Smit et al. “Scaling Multi-Agent Reinforcement Learning to Full 11 versus 11 Simulated Robotic Football”. In: *Autonomous Agents and Multi-Agent Systems* 37.20 (2023). URL: https://doi.org/10.1007/s10458-023-09603-y.

[20] Sphero. *Sphero Bolt*. Accessed: November 28, 2023. Year. URL: https://sphero.com/products/sphero-bolt.

[21] E. Todorov, T. Erez, and Y. Tassa. “MuJoCo: A Physics Engine for Model-Based Control”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012*. Vilamoura, Algarve, Portugal, Oct. 2012, pp. 5026–5033.

[22] A. Wendler and T. Heineken. “ER-Force 2020 Extended Team Description Paper”. In: 2020.

[23] C. Yu and et al. “The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games”. In: *arXiv preprint arXiv:2103.01955* (2021).

[24] N. Zare et al. “CYRUS Soccer Simulation 2D Team Description Paper 2021”. In: *RoboCup 2021: Robot World Cup XXV*. 2021.