Artificial Intelligence Course Project: Multi-Agent Air Hockey using Reinforcement Learning

Aditya Sankhla, Rohit Ashwani, Tushar Bansal, Aditya Dubey

November 21, 2023

1 Introduction and Problem Motivation

The project, "Multi-Agent Air Hockey using Reinforcement Learning," conducted as part of the Artificial Intelligence course at the Indian Institute of Technology Bhilai in the 2023-24M semester, represents a dynamic exploration into the realm of game artificial intelligence. This endeavor embraces the convergence of advanced machine learning techniques, focusing on the application of two distinct algorithms: Proximal Policy Optimization (PPO) and Deep Deterministic Policy Gradient (DDPG).

Motivated by the desire to understand and compare the performance of different reinforcement learning approaches, this project delves into the complexities of the Air Hockey environment within the OpenAI gym. Rather than limiting the investigation to a singular algorithm, our team has ventured into the implementation and analysis of both PPO and DDPG. This dual-algorithm approach allows for a comprehensive examination of the strengths, weaknesses, and adaptability of each method in the specific context of controlling agents in a competitive Air Hockey setting.

PPO, recognized for its simplicity and effectiveness, serves as a benchmark for comparison, while DDPG, known for its applicability in continuous action spaces, introduces a different dimension to our exploration. By embracing a diverse set of reinforcement learning techniques, our project aims to contribute nuanced insights into the challenges and opportunities presented by the Air Hockey environment. The subsequent sections of this report provide a detailed account of our implementations, analyses, and findings for both PPO and DDPG, offering a comprehensive understanding of their respective roles in optimizing agent behavior in the dynamic and competitive world of Air Hockey.

2 Project Details

The hockey environment is a game between two players, player1 and player2. The objective of the game is to maximize the cumulative reward, which is the

sum of rewards over all timesteps. The cumulative reward mainly depends on which agent wins.

To win a game, one agent needs to score a goal. If neither agent scores a goal within 1000 timesteps, the game is automatically drawn. If an agent wins, they receive a reward of 10 in the final timestep, while the losing agent receives a reward of -10. In case of a draw, both agents receive a reward of 0. During each timestep, the agent gets additional reward if they are close to the ball.

The environment has 18 observation dimensions and 4 action dimensions per agent. Each action dimension takes a real value between 0 and 1. Because the action of the opponent is unknown, it is a partially observable setting .

3 DDPG Algorithm Implementation

Reinforcement Learning Scenario

In our setup, the agent interacts with the environment (E) in discrete time steps:

Observes state: x_t
Takes action: a_t
Receives reward: r_t
Policy:
pi (maps states to action probabilities)
Cumulative reward from a state: R_t = $\sum \gamma^{(i-t)} r(s\_i, a\_i)$

After each training step, we retrieve comprehensive statistics, including the number of steps completed, the reward obtained at the end of the episode, and the current epsilon value, among other relevant metrics.

```
Player 2    3: Done after  93 steps.    Reward: -40.677       Eps: 0.81
            4: Done after 100 steps.    Reward: -114.2164     Eps: 0.77
            5: Done after 100 steps.    Reward: -24.4444      Eps: 0.74
            6: Done after 100 steps.    Reward: -15.4121      Eps: 0.7
            7: Done after 100 steps.    Reward: -71.222       Eps: 0.66
Player 2    8: Done after  30 steps.    Reward: -23.7132      Eps: 0.63
            9: Done after 100 steps.    Reward: -100.0289     Eps: 0.6
Player 2   10: Done after  43 steps.    Reward: -35.3312      Eps: 0.57
+--------------+---------+
| Mean reward | -53.882 |
+--------------+---------+
| Mean touch  |   0.3   |
+--------------+---------+
| Mean won    |   0     |
+--------------+---------+
| Mean lost   |   0.4   |
+--------------+---------+
Saving training statistics...
[0.0, 0.0]
            0: Done after 250 steps.    Reward: -6.2241       Touched: 1            Eps: 0
            1: Done after 250 steps.    Reward: -56.3949      Touched: 0            Eps: 0
Player 1    2: Done after  82 steps.    Reward: 6.4067        Touched: 0            Eps: 0
            3: Done after 250 steps.    Reward: -49.8341      Touched: 0            Eps: 0
Player 2    4: Done after 192 steps.    Reward: -19.0879      Touched: 1            Eps: 0
+--------------+---------+
| Mean reward | -25.027 |
+--------------+---------+
| Mean touch  |   0.4   |
+--------------+---------+
| Mean won    |   0.2   |
+--------------+---------+
| Mean lost   |   0.2   |
+--------------+---------+
```

Figure 1: Sample training and evaluation results

Q-Learning Enhancements

To scale up Q-learning, we introduced two key modifications:

- Replay buffer for diverse learning experiences.
- Separate target network for calculating target values (y_t).

Reward Systems

The agent's reward system, based on previous rewards and proximity to the puck, proves beneficial for enhancing performance. By considering historical rewards and real-time proximity, the reinforcement learning model develops nuanced gameplay strategies. Penalties for non-engagement or initial contact incentivize the agent to adopt proactive and skillful behaviors, fostering a learning environment that empowers the agent to refine actions and improve overall performance in air hockey simulations over time.

Actor-Critic with DPG Algorithm

We adopted an actor-critic approach based on the Deterministic Policy Gradient (DPG) algorithm:

Actor: $\mu(s|\theta_\mu)$ (deterministically maps states to actions)
Critic: $Q(s, a)$ (trained using the Bellman equation)
Actor update optimizes expected return: J

Challenges in Continuous Spaces

Exploration in continuous spaces is managed by decaying the probability of random actions.

Neural Network Architecture

Both actor and critic use a simple feedforward neural network:

- Two hidden layers of size 256 with ReLU activation.
- Tanh activation in the actor's readout for actions between -1 and 1.
- Adam optimizer with a learning rate of 0.0001.

Training Details

During each episode:

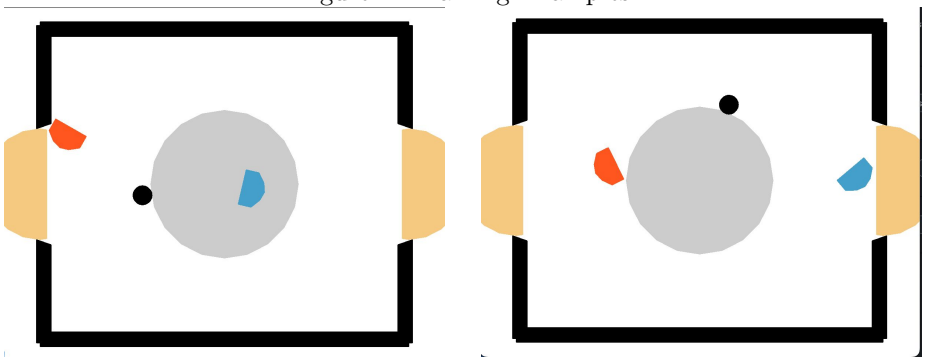
- 30 gradient steps are performed.
- Batch size is set to 128.

Experience replay through the buffer enhances learning stability.

Final Results

After a total of 500 training and 500 evaluation episodes, our experimentation culminated in impressive results. We achieved a final mean reward of 22, a mean touch value of 0.35, and an average of 0.45 victories. Additionally, our reinforcement learning agents demonstrated an astounding win rate of 73%, underscoring the effectiveness of our trained models in mastering the challenges posed by the Air Hockey environment.

Figure 2: Training Examples



The evaluation touch mean graph illustrates an upward trend, indicating improved interaction with the Air Hockey environment.

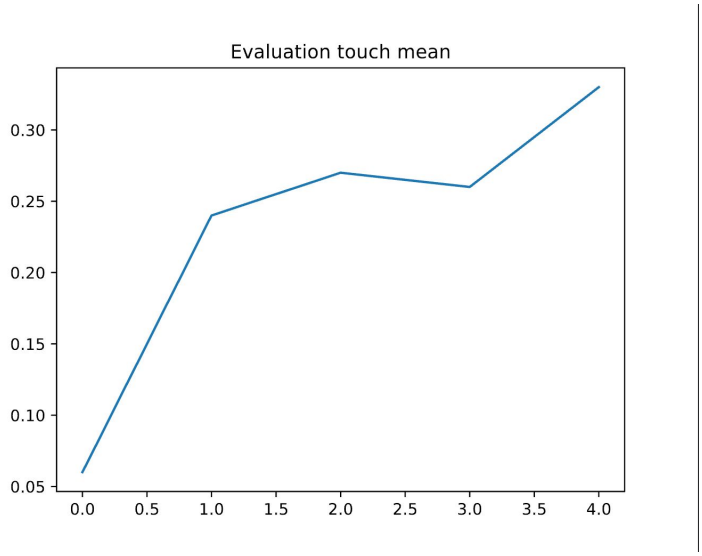


Figure 3: Evaluation Touch Mean

Concurrently, the evaluation reward mean graph showcases a continuous ascent, highlighting enhanced overall performance and strategic decision-making.

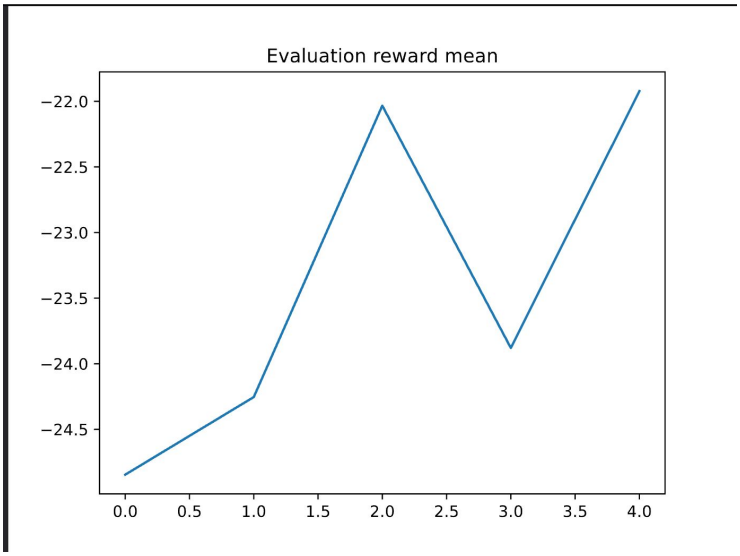


Figure 4: Evaluation Reward Mean

Additionally, the evaluation won vs loss graph tells a compelling success story, depicting our agents consistently outperforming opponents and achieving

a commendable increase in win count.

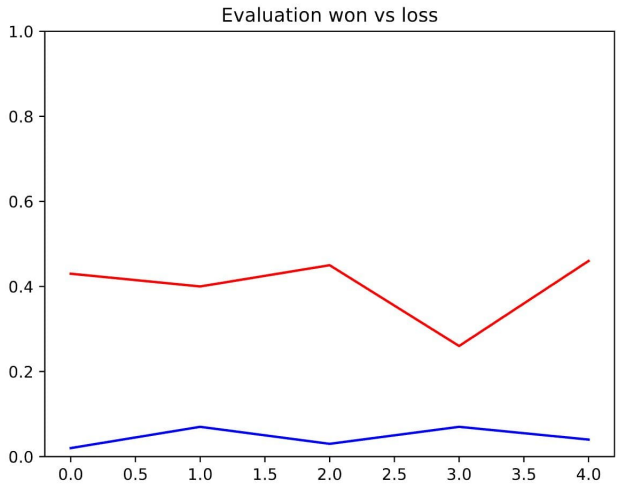


Figure 5: Evaluation Won vs Loss

4 PPO Algorithm Implementation

The PPO (Proximal Policy Optimization) algorithm was used to solve the hockey environment. Since PPO is a policy gradient algorithm, the agent's policy is modeled explicitly.

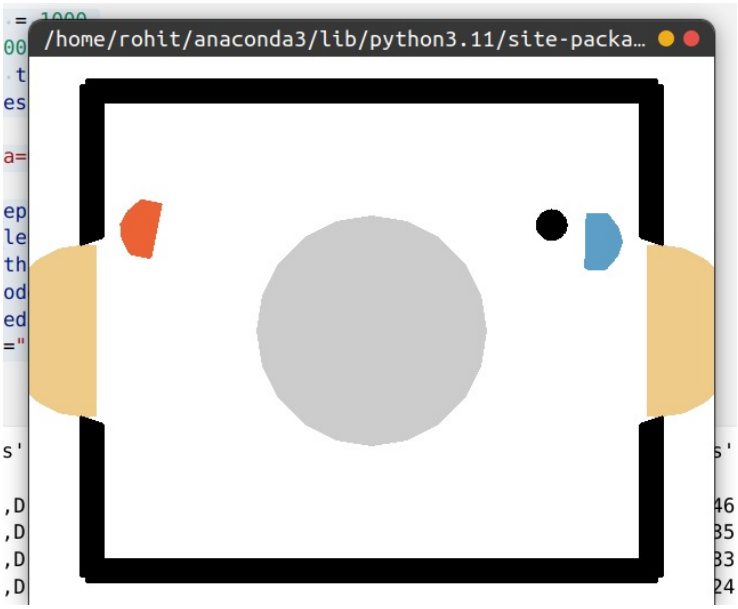


Figure 6: PPO Visualisation Image

Techniques involved

The following techniques were used to implement the algorithm:

1. GAE (Generalized Advantage Estimation) was used as an advantage estimator.
2. A separate network was used for policy and value, but gradient descent was applied using a joint loss function.
3. To transform the actions into a continuous domain, a Multivariate Normal Distribution was used.
4. To ensure that the actions are within [-1,1], *tanh* was used for transformation.
5. Gradient clipping and input normalization were implemented. For normalization, the Welford online algorithm was used.
6. PyTorch was used for the implementation of the neural networks.

Training Details

The whole process of training was split up into alternating periods of training and evaluation. Evaluation was always against the basic opponent and didn't

use exploration. If the agent achieved a record high evaluation performance, his current checkpoint was added to a list of possible opponents. During training, the next opponent (player2) was then sampled from the list of old checkpoints and the basic opponent provided by the laser hockey package. This prevented overfitting of the basic agent.

Final Results

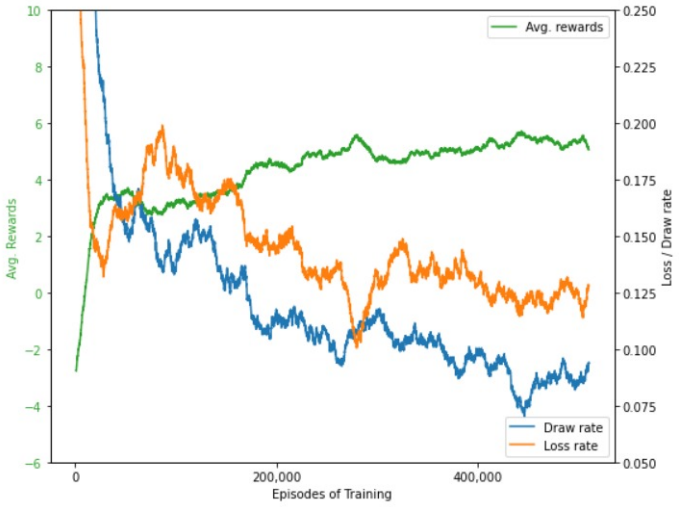


Figure 7: Draw and Loss Rate vs Episodes of Training

- The implementation leads to a win rate of up to **91%** when playing against both opponents in an alternating way. For other hyperparameters the performance might be better.
- Because it is an on-policy algorithm, it is not surprising that beyond a certain point, performance seems to plateau.

5 Project Features

The project has the following features:

1. **DDPG Implementation:** The project incorporates the DDPG algorithm for scenarios with continuous action spaces. DDPG utilizes an actor-critic approach, addressing challenges in exploration through a decaying probability strategy. The modular design enhances scalability, and a user-friendly Jupyter notebook provides implementation guidance.

2. **PPO Implementation:** The project also features the implementation of the PPO algorithm, a model-free, online, on-policy reinforcement learning method. PPO is chosen for its simplicity, good sample efficiency, and robustness to hyperparameters. The modular structure and customizable hyperparameters make it adaptable to different environments.

3. **Modular Design:** The overall project is designed with modularity in mind, ensuring separate files for DDPG and PPO components, algorithms, training, and testing procedures. This design promotes code clarity, simplifies debugging, and facilitates future developments.

4. **Hyperparameter Options:** Users have the flexibility to customize hyperparameters such as the number of epochs, batch size, learning rate, and discount factor. This adaptability is crucial for optimizing performance in the Air Hockey environment.

5. **Easy to Use:** A Jupyter notebook is provided to guide users in implementing and using both DDPG and PPO agents to automate the Air Hockey environment. The interactive guide simplifies the understanding and implementation of reinforcement learning strategies.

6. **Visualization Interface:** The project offers a user-friendly interface for visualizing the training and evaluation processes. This feature is available for both DDPG and PPO, enhancing the user's ability to comprehend the learning progress, debug effectively, and observe agents' behavior in real-time.

6 Conclusion

The DDPG algorithm is a deep reinforcement learning algorithm that uses an actor-critic approach to learn a deterministic policy. The PPO algorithm is another reinforcement learning algorithm that uses a surrogate objective function to update the policy. The project's results showed that the PPO algorithm outperformed the DDPG algorithm in terms of the win rate obtained by the agents.

In conclusion, the project demonstrated that the PPO algorithm is more effective than the DDPG algorithm in the given environment. However, a direct comparision cannot be made and would require extensive training time and resources.

7 References

- **PPO Paper:** https://arxiv.org/abs/1707.06347
- **Pytorch Impementation:** nikhilbarhate99/PPO-PyTorch
- **DDPG understanding** DDPG Implementation article