# Deep Reinforcement Learning on Robotic Agents

Dhrumil Chetankumar Joshi
School of Computing and Augmented Intelligence
Arizona State University
Tempe, AZ, USA
djoshi12@asu.edu

Arshdeep Singh Sachdeva
School of Computing and Augmented Intelligence
Arizona State University
Tempe, AZ, USA
asachd14@asu.edu

Rohith Krishna Papani
School of Computing and Augmented Intelligence
Arizona State University
Tempe, AZ, USA
rpapani@asu.edu

Param Oza
School of Computing and Augmented Intelligence
Arizona State University
Tempe, AZ, USA
puoza@asu.edu

Mary Mccready
School of Computing and Augmented Intelligence
Arizona State University
Tempe, AZ, USA
mkmccre2@asu.edu

## ABSTRACT

The main objective of this assignment was to use Reinforcement Learning algorithms to train agents in reaching their goal. We have worked on the two specified environments in the gymnasium library. We were able to successfully teach the Cartpole agent to balance itself and the humanoid agent to walk. We have successfully recorded rewards for each episode while training. We have trained cartpole-agents using DQN and PPO, and found out that PPO performed the best. We trained the humanoid agent using the SAC, PPO, TD3, A2C and DDPG algorithms [5] and found out that A2C performed the best. In the following report we will show the 3D renders, and the reward vs episode plots of each environment.

## KEYWORDS

Reinforcement Learning, Robotic Agents, Gymnasium, Gym, OpenAI, Stable Baseline 3, MuJoCo

## 1 RELATED WORK

Reinforcement Learning (RL) has proven to be an invaluable approach for training robotic agents, and the integration of OpenAI Gym [3], Stable Baselines3, and MuJoCo[4] has significantly advanced the field. OpenAI Gym serves as a versatile environment for the development and evaluation of RL algorithms, offering a diverse set of simulated tasks that enable robots to learn and adapt. Stable Baselines3 is a user-friendly RL library that simplifies the implementation of various RL algorithms, making it accessible to a wider audience. This ease of use allows researchers and engineers to efficiently train and fine-tune robotic agents within the Gym environment.

Moreover, MuJoCo, a physics engine, adds an essential layer of realism to these simulations by accurately modeling the dynamics and kinematics of robots and their environments. This enhanced realism enables robotic agents to learn complex tasks with precision and efficiency. The combination of these tools has not only accelerated progress in RL research but has also paved the way for the development of robotic agents that can tackle increasingly intricate real-world challenges, making significant contributions to various fields, including robotics, automation, and artificial intelligence.

## 2 PROBLEM FORMULATION

The goal is to utilize Reinforcement Learning (RL) algorithms to train these agents to perform specific tasks. Specifically, the Cartpole agent must learn to balance itself, while the Humanoid agent should learn to walk. Reinforcement Learning (RL) is a subfield of machine learning where an agent interacts with an environment, taking actions and receiving feedback in the form of rewards or penalties. The agent's objective is to learn a policy that maximizes cumulative rewards over time. In this assignment, we will explore the application of RL to two distinct robotic environments provided by the Gym library. The two environments, Cartpole and Humanoid, pose unique challenges and learning objectives.

Stable Baselines3[2], Gymnasium, and MuJoCo are crucial Python packages that will help solve the problem of training robotic agents in the Cartpole and Humanoid environments through Deep Reinforcement Learning (DRL).

### 2.1 Gymnasium

Gymnasium, often referred to as Gym, is an open-source Python library that provides a wide range of pre-built environments for reinforcement learning tasks. In this assignment, Gymnasium plays a pivotal role by offering the Cartpole and Humanoid environments. Gymnasium abstracts the underlying complexity of the environments, making it easier for us to interact with and train agents in these simulated environments. Gymnasium also provides standardized interfaces, making it possible to apply various reinforcement learning algorithms to different problems without having to rewrite the entire codebase. This standardization ensures ease of use.

### 2.2 MuJoCo

MuJoCo is a physics engine often used in reinforcement learning research. It provides a highly efficient and realistic simulation of physical environments. MuJoCo offers a more realistic simulation of the dynamics and physics of the robotic environments. This realism is essential for training agents that need to learn complex motor control tasks, such as balancing the Cartpole or walking for the Humanoid. MuJoCo is known for its efficiency, which is crucial when dealing with complex robotic environments. Efficient simulation allows for faster training of reinforcement learning agents.

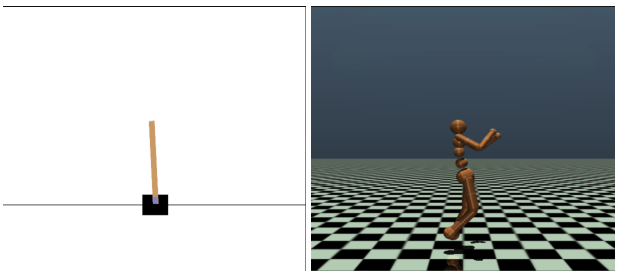

**Figure 1: Cartpole and Humanoid Gymnasium Environments [1]**

## 2.3 Stable Baseline3

Stable Baselines3 is a high-level library built on top of OpenAI's Baselines. It provides a user-friendly interface for training and evaluating reinforcement learning agents. Stable Baselines3 offers various state-of-the-art reinforcement learning algorithms, including PPO (Proximal Policy Optimization), A2C (Advantage Actor-Critic), DDPG (Deep Deterministic Policy Gradient), and more. Stable Baselines3 provides a well-structured and easy-to-use infrastructure for training agents. This simplifies the process of training agents using different RL algorithms. It is designed to be reproducible, which is essential for research and academic purposes. This ensures that the results obtained by different groups using the library can be compared and validated.

# 3 METHODOLOGY & ASSUMPTIONS

We trained each RL algorithm for 1 Million time steps. We used MlpPolicy for each of our algorithm that we used from Stable Baseline3. We will now discuss our approach in two main environments, namely Cartpole and Humanoid.

## Cartpole Environment

For the Cartpole environment, we will present two primary algorithms:

*3.0.1 Deep Q-Network (DQN): .* DQN is a deep reinforcement learning algorithm commonly used for solving tasks in discrete action spaces. It is a type of Q-learning that employs deep neural networks to approximate the Q-value function, which estimates the expected cumulative rewards for taking actions in a given state. DQN is known for its robustness and has been applied in various domains, including game playing and control tasks. It's particularly useful for tasks where there's a finite set of discrete actions, and the agent needs to learn the optimal action-value function.

It was the chosen algorithm because of its strong performance in discrete action spaces. The task of balancing the cartpole involves making binary decisions (push left or push right), which aligns perfectly with DQN's strengths. DQN's ability to handle discrete actions efficiently and its robustness in training made it a suitable candidate for our Cartpole experiment. We aimed to leverage DQN's capability to find the optimal action-value function and successfully teach our agent to balance the pole on the cart.

*3.0.2 Proximal Policy Optimization (PPO): .* PPO is a popular reinforcement learning algorithm that focuses on optimizing policies for continuous action spaces. It is designed to work with environments where the agent interacts with the environment and learns how to perform a task by observing its actions' consequences.

In PPO, the agent iteratively improves its policy by collecting experience data, estimating the advantages of different actions, and updating its policy to maximize expected rewards. One of PPO's defining characteristics is its "proximal" update rule, which limits large policy changes to ensure training stability. PPO is particularly useful in environments where smooth policy updates are preferred, preventing abrupt changes that can hinder learning.

PPO was our choice for the Cartpole experiment due to its simplicity, training stability, sample efficiency, and strong overall performance. It offers straightforward implementation, reduces policy disruptions, and requires fewer interactions with the environment for learning. These qualities make PPO a reliable and efficient algorithm for teaching our agent to balance the pole on the cart in the Cartpole environment.

## Humanoid Environment

In the Humanoid environment, we will explore five different algorithms:

*3.0.3 Proximal Policy Optimization (PPO).* For our Humanoid walking experiment, PPO was selected as the reinforcement learning algorithm due to its versatility and effectiveness. The Humanoid agent needed to learn to walk, a task that involves continuous actions and a complex action space. PPO's stability, sample efficiency, and policy optimization made it an excellent choice for this high-dimensional, challenging task. The goal was to leverage PPO's ability to balance exploration and exploitation effectively, allowing the Humanoid agent to learn a robust walking policy. While other algorithms may excel in specific areas, PPO's combination of attributes made it a strong candidate for teaching the Humanoid agent to walk in our experiment.

*3.0.4 Soft Actor-Critic (SAC).* The Soft Actor-Critic (SAC) algorithm is an off-policy actor-critic deep reinforcement learning method designed for continuous action spaces. SAC is particularly noted for its stability and efficiency. It integrates the idea of maximum entropy reinforcement learning into the standard actor-critic framework, thereby promoting exploration and more robust learning.Soft Actor-Critic (SAC) is an advanced deep reinforcement learning algorithm that can be employed to train humanoid robots to walk. By representing the robot's state (like joint positions) and actions (like motor torques) and using a defined reward function to guide desired behaviors, SAC allows the robot to explore and learn from its environment. The algorithm's off-policy nature utilizes a replay buffer, enabling the robot to learn from past experiences, while its continuous action space ensures smooth control, making it apt for the complexities of humanoid walking. By iterating through this learning process, the robot refines its walking patterns, becoming more adept and stable over time.

Here's a high-level breakdown of SAC:

**Maximum Entropy RL Framework:** Unlike traditional RL where the objective is to maximize the expected reward, in maximum entropy RL, the objective is to maximize both expected reward and the entropy of the policy. This encourages a stochastic policy and ensures more exploration. The objective function is:

$$J(\pi) = \mathbb{E}_{s\sim\rho, a\sim\pi}\left[R(s,a) + \alpha \log \pi(a|s)\right]$$

where $\rho$ is the state-visitation distribution, $R(s,a)$ is the reward, $\alpha$ is a temperature parameter that weighs the importance of entropy against the reward, and $\pi$ is the policy.

**Soft Value Function:** The Q-function and Value function are redefined with the entropy term:

$$Q^{\pi}(s,a) = \mathbb{E}\left[R(s,a) + \gamma \mathbb{E}\left[V^{\pi}(s') - \alpha \log \pi(a'|s')\right]\right]$$

$$V^{\pi}(s) = \mathbb{E}_{a\sim\pi}\left[Q^{\pi}(s,a) - \alpha \log \pi(a|s)\right]$$

where $\gamma$ is the discount factor.

**Training:**

- Sample a batch of transitions from the replay buffer.
- Update the Q-function using the Bellman equation with a soft target update.
- Update the policy network to maximize $J(\pi)$.
- (Optionally) Adjust the temperature parameter $\alpha$.
- Use soft updates to gradually move the target Q-network towards the current Q-network.

*3.0.5 Advantage Actor-Critic (A2C).* A2C is a reinforcement learning algorithm that combines the advantages of both actor-critic methods and advantage estimation. It is typically used in tasks with both discrete and continuous action spaces. A2C is known for its properties such as stability, efficiency, and scalability.

In A2C, the actor network chooses actions based on the current policy, while the critic network evaluates those actions by estimating their advantage over other possible actions in a given state. The advantage function measures the difference between the actual outcome and the expected outcome, helping the agent to focus on actions that lead to better-than-expected results. This combination of actor and critic networks allows A2C to balance exploration and exploitation effectively.

For our Humanoid walking experiment, we selected A2C as it is a versatile algorithm that can handle tasks with both discrete and continuous action spaces, making it suitable for teaching the Humanoid agent to walk. Its stability and efficiency are valuable in complex, high-dimensional environments like humanoid walking, and the advantage estimation helps the agent make better action choices. While A2C might not be as sample-efficient as some other algorithms, its overall performance and reliability in various domains make it a strong candidate for the Humanoid experiment, allowing us to explore its potential for training the agent in a challenging task.

*3.0.6 Deep Deterministic Policy Gradient (DDPG).* DDPG is a reinforcement learning algorithm designed to address tasks with continuous action spaces. It combines elements from Deep Q-Networks (DQN) and the deterministic policy gradient to enable learning in such environments. DDPG is commonly used in scenarios where the agent needs to learn a policy for continuous actions. Its key properties include an actor-critic architecture, deterministic policy, experience replay, target networks, and the use of the Bellman equation.

In our Humanoid experiment, we chose DDPG to teach the Humanoid agent to walk for several reasons. The Humanoid task requires precise control over a wide range of continuous actions, making DDPG a suitable choice due to its deterministic policy that can handle such actions effectively. Moreover, DDPG's use of experience replay and target networks can stabilize learning in a complex, high-dimensional environment like the Humanoid task. However, it's essential to note that DDPG's performance in our experiment was lower compared to other algorithms, which may be attributed to challenges related to exploration efficiency, sample efficiency, and hyperparameter sensitivity. Despite this, DDPG was selected for its potential in addressing continuous action tasks like humanoid walking, and the experiment aimed to explore its capabilities and limitations in this context.

*3.0.7 Twin Delayed Deep Deterministic Policy Gradient (TD3).* The final subsection will cover the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm and its use in the Humanoid environment.

TD3 being an actor critic algorithm has an actor network that learns the policy and a critic network which learns the value function. TD3 specifically learns a policy to maximize expected cumulative reward in a continuous environment.The TD3 algorithm is a unique algorithm as it has Twin critics to estimate value of each state which helps with the stability of the program by avoiding overestimation. It is employing a technique called clipped double Q-learning which chooses the lower of two estimates provided by the two critic networks.

We used the TD3 algorithm in the humanoid environment citing the reason for it being an actor critic algorithm which is favorable to solve the continuous control tasks. The results given by TD3 algorithm on the basis of the rendered 3D video and rewards vs episode graph suggested otherwise. The reward vs episode was almost flat after one point which suggests that agent was not making any progress in the environment. Training an humanoid agent might be a high dimensional task which could be one potential reason for the under-performance of TD3 as it might not be able to generalize all possible states.

# 4 RESULT AND ANALYSIS

## 4.1 Cartpole results

In the Cartpole experiment, we conducted a comparison between the performance of the Deep Q-Network (DQN) and Proximal Policy Optimization (PPO) algorithms. The training progress was monitored across timesteps to evaluate their effectiveness in mastering the binary task of balancing the cartpole in the Cartpole v1 environment.

As illustrated in the figure depicting the training progress, a clear trend emerged. Over time, it became evident that PPO consistently outperformed DQN in terms of achieving better average results. As we progressed through the timesteps, PPO displayed a higher

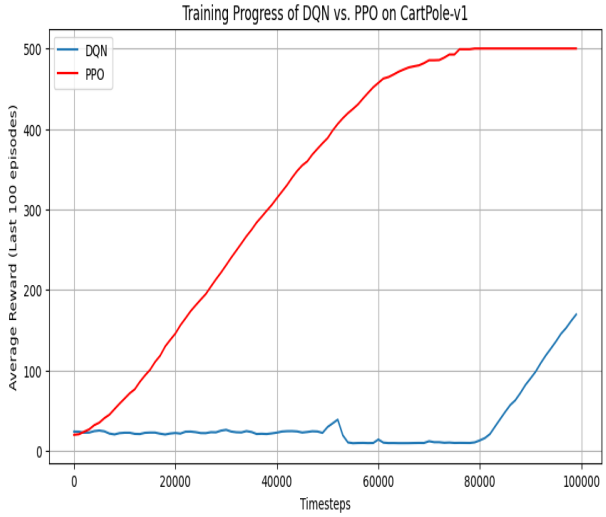


**Figure 2: Reward V/S Time Steps Graph of Cartpole**

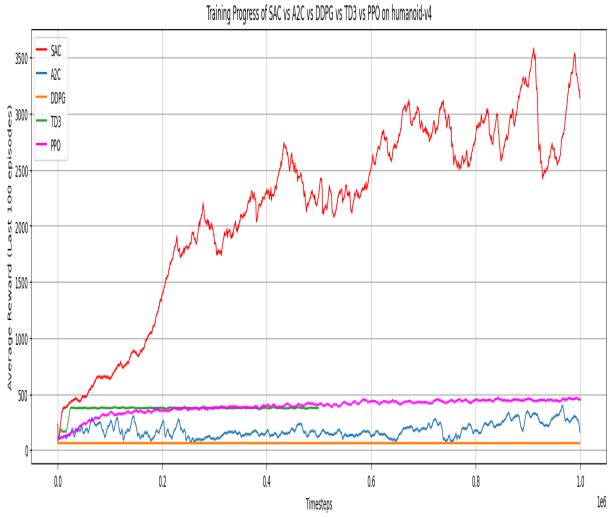


**Figure 3: Reward V/S Time Steps Graph of Cartpole**

degree of proficiency in maintaining the cartpole in an upright position.

PPO's superior performance in the Cartpole experiment can be attributed to its suitability for continuous action control, nuanced policy updates, and stable learning. PPO's capacity to fine-tune continuous actions aligns well with Cartpole's dynamics, while its controlled policy adjustments contribute to a stable training process, leading to better outcomes.

These results underline the efficacy of PPO in mastering the Cartpole task, emphasizing its superiority over DQN in achieving better average performance throughout the training process. The figure's data provides valuable insights into the strengths of PPO for this specific task.

## 4.2 Humanoid results

In the Humanoid experiment, we evaluated the performance of five reinforcement learning algorithms: Soft Actor-Critic (SAC), Advantage Actor-Critic (A2C), Deep Deterministic Policy Gradient (DDPG), Twin Delayed Deep Deterministic Policy Gradient (TD3), and Proximal Policy Optimization (PPO). The training progress of these algorithms was tracked over time to assess their efficacy in teaching the Humanoid agent to walk.

The accompanying figure vividly illustrates the training progress. Notably, SAC stands out as the frontrunner, demonstrating substantially superior performance compared to the other algorithms. PPO and TD3 appear below SAC and exhibit relatively similar progress. A2C follows as the second-to-bottom performer, with DDPG situated at the bottom of the chart.

This hierarchy in performance can be attributed to various factors. SAC's remarkable success can be attributed to its superior handling of continuous action spaces, efficient exploration strategy, and robustness in learning complex policies. PPO and TD3, while competent, fall slightly behind SAC due to differences in policy optimization and exploration. A2C, although effective in some contexts, lags behind the others in this particular task. DDPG's position at the bottom is indicative of its challenges in exploration efficiency, sample efficiency, and hyperparameter sensitivity, which affect its suitability for the Humanoid walking task. These results highlight the strengths and limitations of each algorithm in the context of teaching the Humanoid agent to walk effectively.

## 5 CONCLUSION

In summary, we can say that our approach looks to combine the best of both worlds. It uses the asynchronous approach to gain speed and synchronous approach for accuracy. And the switch from asynchronous to synchronous is made using a threshold function which takes the learning rate into account for determining how many parameters get aggregated for synchronization. Based on the extensive experimentation, we can draw the conclusion that for the same time period, our approach leads to better accuracy and lower loss than both synchronous and asynchronous approaches. As batch size decreases, which is the norm for large training datasets, our approach performs even better. The step size is currently defined in terms of the learning rate and even if the step size is not selected appropriately, our approach will give better results than the synchronous approach.

## REFERENCES


[1] Cart pole gymnasium. https://gymnasium.farama.org/environments/classic_control/cart_pole/.

[2] Dlr-Rm. Dlr-rm/stable-baselines3: Pytorch version of stable baselines, reliable implementations of reinforcement learning algorithms. https://github.com/DLR-RM/stable-baselines3.

[3] Farama-Foundation. Farama-Foundation/Gymnasium: An API standard for single-agent reinforcement learning environments, with popular reference environments and Related Utilities (formerly gym). https://github.com/Farama-Foundation/Gymnasium, no date. Accessed: 15 October 2023.

[4] Google-Deepmind. Google-deepmind/mujoco: Multi-Joint Dynamics with Contact. A General Purpose Physics Simulator. https://github.com/google-deepmind/mujoco, no date. Accessed: 15 October 2023.

[5] RL-Algorithms. Stable baseline3 rl algorithms. https://stable-baselines3.readthedocs.io/en/master/guide/algos.html.