Reinforcement Learning for Robotic Control in Simulated Environments - Bonus Project

Farid Karimi - 401222112

August 20, 2025

Abstract

This report details an Project into reinforcement learning techniques applied to simulated robotic manipulation. The study uses the Panda-Gym environment to train an agent for two distinct tasks: a foundational reaching task and a more involved pushing task. The project began with an exploratory analysis of the environment's mechanics. Subsequently, a Deep Deterministic Policy Gradient agent was trained for the reaching task, followed by a comparative training of both DDPG and Soft Actor-Critic agents for the pushing challenge. The effectiveness of the learned behaviors was then assessed through quantitative success metrics. The results confirm that the agents could successfully learn policies for both tasks but it also underlines the fact that panda-reach is a lot more easier to learn than panda-push which needed a lot of training and still its capabilities is far less than what it could have been which is evidenced by their learning curves and performance in evaluation trials.

1 Introduction

Simulated environments provide a valuable and practical setting for developing robotic control systems without the risks and costs associated with physical hardware. This project is centered on the Panda-Gym simulation, a platform designed for reinforcement learning research. The work was undertaken to build a practical understanding of reinforcement learning by training an agent to solve robotic tasks of varying difficulty. The investigation starts with a straightforward reaching task to establish a baseline. It then progresses to a more complex pushing task, which requires more nuanced interaction with objects in the environment. This document outlines the methods used to explore the simulation, train the learning agents, and evaluate their performance.

2 Environment and Methods

The first part of the project involved becoming familiar with the *PandaReach-v3* environment. In this scenario, the robot's objective is to move its gripper to a target position that is randomly set at the start of each trial. The agent receives information about the gripper's 3D coordinates, the target's 3D coordinates, and the robot's joint states. The agent's actions are controlled by a three-value vector that dictates movement along the x, y, and z axes. To understand the system's behavior, an initial experiment was conducted where the robot arm performed random actions for a few episodes. This provided a qualitative feel for how the robot responded to different inputs.

After this initial exploration, a Deep Deterministic Policy Gradient (DDPG) agent was trained to master the reaching task. The agent's learning was guided by a dense reward signal, which provides continuous feedback based on the proximity of the gripper to the target. This approach is intended to make the learning process more efficient. The agent was trained over 25,000 timesteps, and its progress was tracked by plotting the rewards it received. The learning curve shows a steady increase in rewards, which suggests the agent successfully learned an effective policy.

3 Advancing to a More Complex Task

The project then transitioned to the more demanding *PandaPush-v3* environment. The goal here is to move a block to a specific target location. This task adds a layer of complexity as it requires the agent to interact with an object to achieve its goal. The DDPG agent architecture was first applied to this new

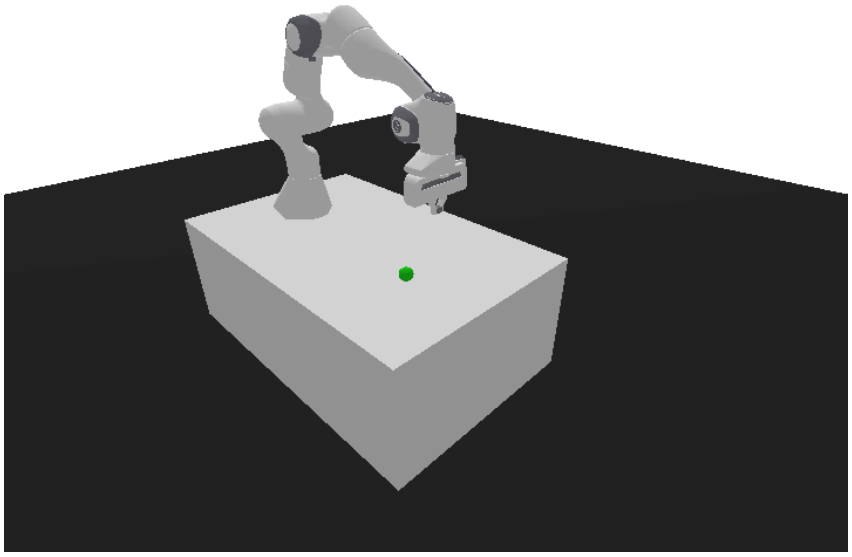

Figure 1: A visual sequence of the robotic arm executing random movements within the *PandaReach-v3* environment.

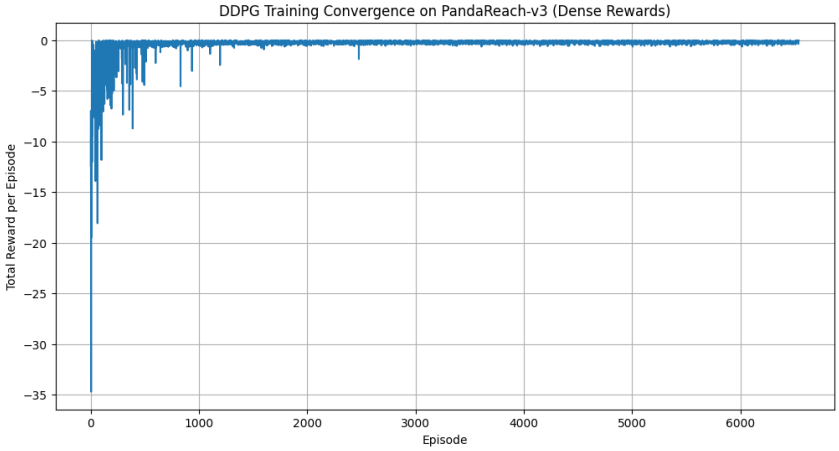

Figure 2: The learning curve for the DDPG agent on the *PandaReach-v3* task, showing episodic rewards over the training period.

challenge, with training conducted over 50,000 time steps. Following that, a Soft Actor-Critic (SAC) agent, another type of reinforcement learning algorithm, was also trained on the same task for 500,000 time steps to compare its performance, it is also important to mention that based on My research on Stack-overflow threads the model needs a couple millions of steps at least to be able to get Better results since this project documentation was not provided in an earlier rime frame and I as a student in Iran do not have access to powerful GPUs I let it train for 3-4 hours on the googlecolab T4 Gpu and the results are a lot better than the first attempt but still the robot lacks dexterity in its movement. based on research on various projects similar to this one I came to conclusion that the DDPG method is not the best when encountering such tasks, it is also important to mention that Both agents showed positive learning trends, indicating their ability to develop strategies for the pushing task.

4 Performance Analysis

To measure the agents' capabilities, the trained models for both the reaching and pushing tasks were evaluated. The models were tested in 50 new episodes with randomized starting conditions to assess their ability to generalize. For the *PandaReach-v3* task, the DDPG agent was successful in nearly all attempts. The evaluation of the DDPG model for the *PandaPush-v3* task also showed a respectable success rate,

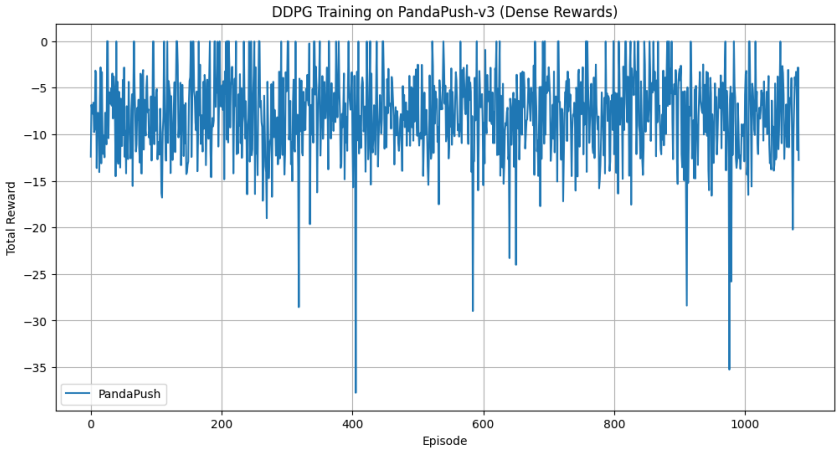

Figure 3: The training progression for the DDPG agent on the *PandaPush-v3* task.

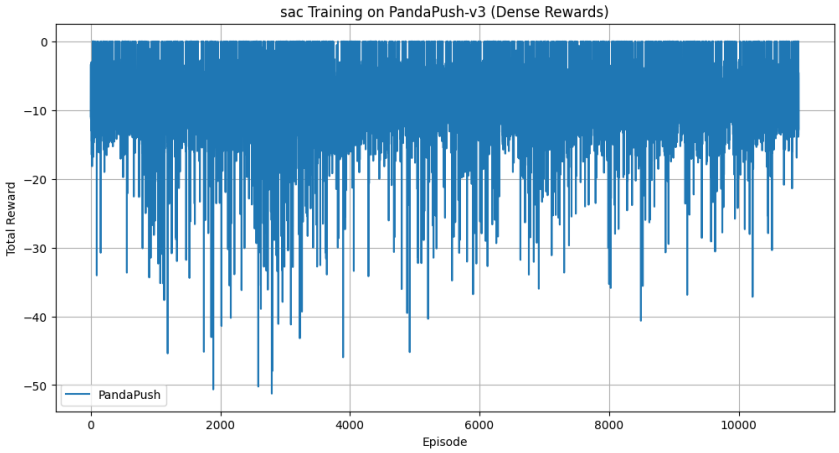

Figure 4: The learning progression for the SAC agent on the *PandaPush-v3* task.

confirming that the agent learned a viable strategy for pushing the block. These quantitative results provide a clear measure of the policies learned during training.

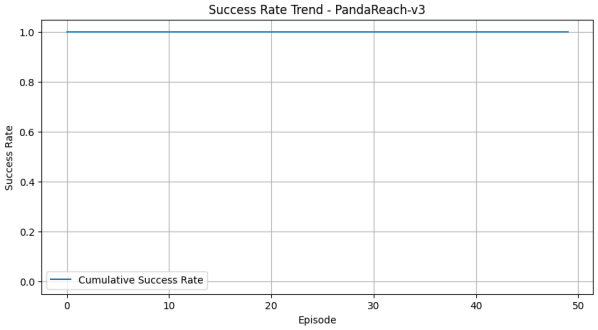

Figure 5: The cumulative success rate of the DDPG agent over 50 evaluation episodes for the *PandaReach-v3* task.

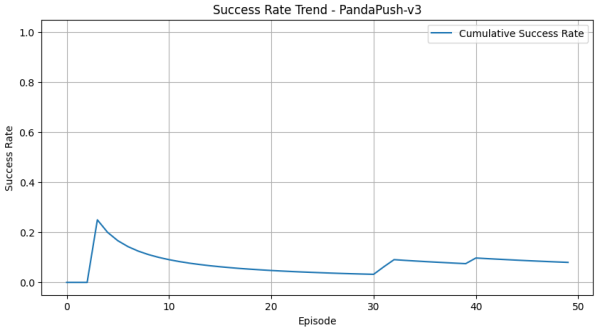

Figure 6: The cumulative success rate of the DDPG agent over 50 evaluation episodes for the *PandaPush-v3* task.

5 Conclusion

This project offered a practical application of reinforcement learning for robotic control within the Panda-Gym simulation. The DDPG agent demonstrated its capacity to learn policies for both a basic reaching task and a more intricate pushing task which could be imporved with longer training. The comparison with the SAC agent on the pushing task provided insight into how different algorithms might approach the same problem. The final evaluations, based on success rates, confirmed that the training process resulted in effective and generalizable behaviors and also How good the algorithm for Reaching task workes since the success rate is always True and never missed in the entire 50 episode it has been tested therefor th is study build confidence in using these methods for developing control solutions in simulated robotic environments.