Task Oriented Reinforcement Learning for Robotic Arm Manipulation: A Comparative Study

BTech. Computer Science Engineering (spl. with AI & DS)

School of Computing

SASTRA Deemed University

Venkatasundaram S

125156131@sastra.ac.in

Manoj Kumar D

125156068@sastra.ac.in

1. Abstract

The task of effectively training robotic arms, for complex manipulation in real-world settings remains challenging due to the high potential for damage, both to the equipment and the environment. To address this, our study leverages simulation through the PyBullet framework as a cost-effective, risk-free environment for evaluating reinforcement learning (RL) algorithms across three primary manipulation tasks: reaching, pushing, and pick-and-place. We tested three off-policy methods- Deep Deterministic Policy Gradient (DDPG), Soft Actor-Critic (SAC), and Twin Delayed DDPG (TD3) and two on-policy methods- Proximal Policy Optimization (PPO) and Advantage Actor-Critic (A2C). To optimize performance, we incorporated Hindsight Experience Replay (HER) with off-policy methods and Generalized Advantage Estimation (GAE) with on-policy methods. Our findings suggest that off-policy algorithms generally outperform on-policy methods for robotic manipulation, especially when aided by HER. The stability and enhanced learning efficiency of off-policy algorithms make them particularly suited for scenarios requiring sample efficiency and continuous, accurate control.

Keywords: Reinforcement Learning, Robotic Arm Manipulation, Deep Deterministic Policy Gradient(DDPG), Soft-Actor-Critic(SAC), Twin Delayed Deep Deterministic Policy Gradient(TD3), Proximal Policy Optimization (PPO), and Advantage Actor-Critic (A2C)

2. Introduction

Robotic manipulation tasks, such as reaching, pushing, and pick-and-place, pose considerable challenges in dynamic environments due to the precision and adaptability required. Reinforcement Learning (RL), an increasingly prominent method for training robots, provides a way to develop flexible policies through trial and error. [13] However, applying RL directly to physical robotic systems can be impractical due to the risk of physical damage, high costs, and limited experimentation capabilities. Instead, training these algorithms in a simulation environment like PyBullet offers a safer, more efficient approach for developing and refining manipulation policies. The cost-effectiveness and accessibility of PyBullet have made it a preferred simulation framework over alternatives like MuJoCo, especially for researchers working on reinforcement learning in robotics [14].

Our project aims to find the most effective RL policies for robotic manipulation tasks using the Franka Emika Panda robotic arm. The tasks focus on end-to-end control, where the robot must autonomously learn to reach an object, push it along a designated path, or execute a pick-and-place sequence. The simulation's inputs include the robot's initial state and target goals, while the outputs measure the policy's success rate in achieving the desired manipulation.

To achieve optimal task performance, we tested three off-policy algorithms- DDPG, SAC, and TD3 and comparing their effectiveness against two on-policy methods, PPO and A2C. Off-policy methods typically allow agents to learn from both past and recent experiences, independent of the policy being followed, making them more sample-efficient. By storing diverse transitions, these methods significantly reduce the sample inefficiency associated with on-policy approaches, which rely solely on immediate experiences within a fixed policy. Furthermore, incorporating HER allows off-policy methods to generate additional, goal-directed experiences from unsuccessful

attempts, transforming them into productive learning episodes. In complex manipulation tasks, this replay mechanism allows the agent to learn not just from successes but from failures too, making it particularly advantageous for sparse-reward environments. Conversely, on-policy methods can be less adaptable for manipulation tasks, as they rely heavily on frequent policy updates, potentially resulting in higher variance in learning and less stability.

Our study underscores the value of off-policy methods with HER, which consistently yield smoother convergence and superior task performance in robotic manipulation. Through this research, we aim to provide insights into selecting suitable RL algorithms for robotic arms in simulation and real-world applications, with a particular focus on task complexity and adaptability.

3. Related Work

Research on reinforcement learning for robotic manipulation has evolved significantly, with various approaches designed to optimize performance for complex tasks. The foundational Deep Deterministic Policy Gradient (DDPG) algorithm, proposed by Lillicrap et al., enables continuous control in robotic environments by combining actor-critic methods with deterministic policies for smoother control [1]. While effective, DDPG's sensitivity to hyperparameters and reward sparsity often hinders performance in tasks requiring precise, multi-step actions. Building on this, Soft Actor-Critic (SAC) introduces entropy maximization for enhanced exploration, making it robust for tasks with dynamic environments [2]. SAC has been further extended in [22], showcasing its ability to handle complex energy-based policies that optimize for both exploration and exploitation in manipulation tasks. However, SAC's stochastic approach sometimes sacrifices fine-tuned control, which may affect performance in precision tasks. Similarly, Fujimoto et al.'s Twin Delayed Deep Deterministic (TD3) improves DDPG by addressing function approximation errors, providing more stable learning, though it can be challenging to tune across varied task complexities[3].

To counter sparse reward signals in complex environments, Andrychowicz et al. developed Hindsight Experience Replay (HER), a method that retrospectively treats failures as alternative goals, significantly improving sample efficiency for off-policy algorithms [4]. This technique has proven beneficial for tasks like robotic manipulation, but its reliance on off-policy methods limits its applicability to algorithms like SAC, DDPG, and TD3.

On-policy methods have also been adapted for robotic tasks, with Proximal Policy Optimization (PPO) by Schulman et al. being particularly notable. PPO stabilizes policy updates by using clipping, making it suitable for environments with large observation spaces [5]. Proximal Policy Optimization (PPO), introduced in [20], represents a breakthrough in on-policy RL methods, offering a stable and efficient framework for continuous control tasks, including robotic manipulation. However, PPO's limitations in handling long-horizon tasks without frequent positive feedback can reduce efficiency in sparse reward tasks. A3C, a simpler on-policy method, utilizes asynchronous updates, which reduce training time but often leads to suboptimal performance in highly complex tasks due to lower sample efficiency [6]. Generalized Advantage Estimation (GAE) is employed alongside PPO and A2C to manage the variance-bias trade-off, improving policy gradient methods' convergence rates.

Further enhancements in robotic manipulation have been achieved by QT-Opt, a Q-learning extension developed for vision-based tasks, integrating visual perception directly into control policies. Despite strong performance in object-oriented tasks, QT-Opt's reliance on image-based training increases computational requirements and limits real-time applications [7]. Additionally, methods like Prioritized Experience Replay (PER) select crucial experiences during training, which improves convergence speed in off-policy algorithms but increases memory consumption [8][9]. Temporal Difference Models, as proposed in [19], offer an alternative approach for handling model-based and model-free RL challenges, providing insights into balancing sample efficiency with policy performance.

Meta-learning, as explored by Duan et al. in RL^2, applies reinforcement learning to enable agents to quickly adapt to new environments by training them on multiple tasks beforehand [11]. Although powerful for multi-task environments, meta-learning's high computational demands and need for extensive pre-training data make it less practical for specific robotic tasks requiring precision. Sim-to-real transfer remains a key challenge in RL for

robotics, where policies trained in simulation often fail to generalize well to real-world scenarios. Works such as [15] highlight how simulation environments like PyBullet support domain randomization and other techniques for better generalization.

Finally, OpenAI's Dactyl project demonstrates domain randomization to enhance simulation-to-real-world transfer by training on diverse simulated conditions. This approach aids in building resilient policies that adapt to real-world variability; however, its requirement for significant computational resources limits accessibility for small-scale implementations [10]. A critical analysis of RL algorithms for robotic manipulation was presented in [16], emphasizing the significance of proper benchmarking metrics to evaluate algorithmic performance consistently.

Our study builds on these works by comparing both off-policy methods (DDPG, SAC, TD3) with HER for complex robotic tasks and on-policy approaches (PPO, A2C) with GAE. By leveraging the strengths of each and addressing limitations—such as sensitivity to hyperparameters in off-policy algorithms and instability in on-policy methods—our work seeks to optimize training efficiency for the challenging tasks of reaching, pushing, and pick-and-place in robotic manipulation.

4. Dataset and Features

In this project, we simulate a Franka Emika Panda robotic arm to execute manipulation tasks like reaching, pushing, and pick-and-place in the PyBullet environment. The robotic arm and task environment are defined in Universal Robot Description Format (URDF) files, specifying the mechanical structure and properties of the arm and target objects within the simulation.

4.1 Environment and Action-Space Setup :

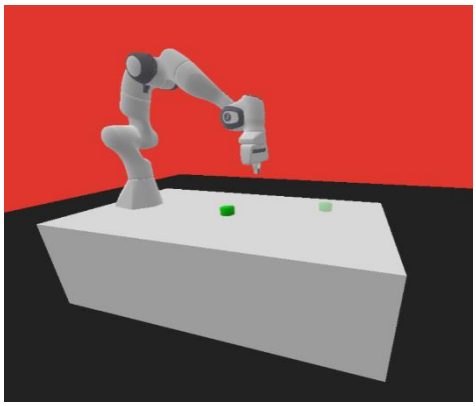


Figure 4.1 : Franka Emika Panda environment

The robotic arm operates with 7 Degrees of Freedom (DOF), plus an additional actuator for the gripper, enabling it to perform complex tasks with flexibility and precision. The action space consists of continuous joint angles or Cartesian coordinates, depending on the specific task, allowing fine-grained control over the arm's positioning and movement.

4.2 State Representation and Observations :

Each state vector is composed of joint positions, joint velocities, the Cartesian position of the end effector, and the relative position to the target object. Additionally, for tasks involving grasping or pushing, the state includes the target's position. This setup provides a comprehensive set of features necessary for precise robotic manipulation.

4.3 Reward Structure :

The reward system is designed to guide the agent efficiently toward successful task completion. Initially, the reward is set to -50, encouraging the agent to take action to reduce this penalty by making progress. For each step, a small penalty of -0.01 is subtracted, incentivizing the agent to complete tasks in fewer steps. Incremental rewards are given as the arm gets closer to the target, promoting gradual but focused movement. Upon successful completion of a task, the agent receives a reward boost, with typical values set at +1 or +1.5 depending on task complexity. This sparse but well-structured reward scheme balances exploration with goal-oriented actions, allowing the agent to learn effective task strategies over time.

4.4 Termination Conditions :

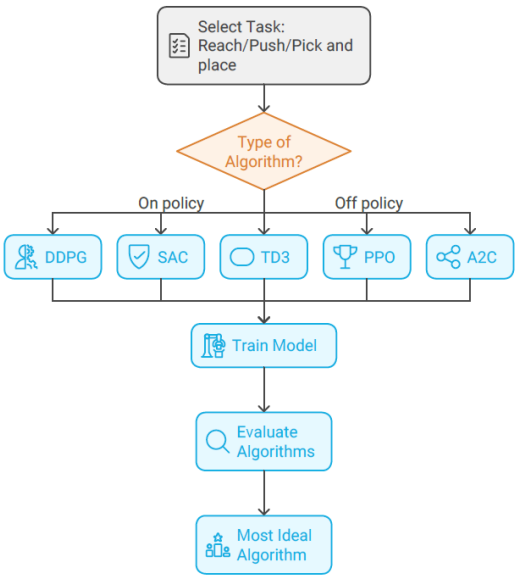
An episode terminates upon task completion, failure to achieve the task within a specified timestep limit, or if the arm's movements violate predefined spatial boundaries. This helps the agent learn boundary conditions and task constraints, contributing to safe and reliable policy learning.

4.5 Preprocessing and Normalization :

Normalization is applied to rewards across episodes, ensuring that varying episode lengths or reward magnitudes do not impact learning stability. Action and state vectors are also scaled within the environment to enhance stability and make learning smoother for the agent, particularly beneficial for manipulation tasks where precision is crucial.

5. Methods

The reinforcement learning (RL) algorithms used in this study are categorized into off-policy and on-policy approaches. Each algorithm has been selected based on its unique capabilities in handling continuous action spaces and sparse-reward environments, as seen in robotic manipulation. We present each algorithm's mathematical foundation, elaborate on actor and critic components, and discuss the roles of HER and GAE in improving learning efficiency and stability.



5.1 Off-Policy Methods :

Off-policy methods allow for more efficient sample reuse by training on past experiences, even if these were generated by previous versions of the policy. This is particularly beneficial for complex robotic manipulation tasks, where generating new data can be costly.

5.1.1 Deep Deterministic Policy Gradient (DDPG)

DDPG extends the deterministic policy gradient algorithm for continuous action spaces, combining elements from Q-learning and actor-critic architectures to optimize policy and value functions simultaneously.

- **Actor Network** : The actor network in DDPG, denoted by $\mu(s|\theta^{\mu})$ outputs continuous actions directly based on the current state (s) .Unlike traditional policies that select actions probabilistically, the actor network deterministically maps each state to a specific action, which is critical in applications like robotics that require precision.
- **Critic Network** : The critic, $(Q(s,a|\theta^{Q}))$, estimates the action-value function, giving a numerical score for each action taken in a particular state. This score indicates the expected future reward. The critic's objective is to minimize the temporal difference (TD) error, given by:

$$L(\theta^{Q}) = E_{(s,a,r,s')}\left[\left(r + \gamma Q(s', \mu(s'|\theta^{\mu})|\theta^{Q}) - Q(s,a|\theta^{Q})\right)^2\right]$$

  where γ is the discount factor. The TD error guides the critic to more accurately assess action values over time, providing a learning signal to update the actor.

By working together, the actor learns a policy that maximizes the expected cumulative reward, while the critic evaluates actions to refine the policy.

5.1.2 Soft Actor-Critic (SAC)

SAC differs from DDPG in its stochastic approach, aiming to maximize both cumulative reward and policy entropy. This entropy component encourages the agent to explore various actions rather than prematurely converging to suboptimal ones, which can be valuable in complex robotic tasks.

- **Actor (Policy) Network** : The SAC actor produces a probability distribution over actions rather than a single deterministic action, represented as $\pi(a|s)$. This encourages exploration. By sampling from this distribution, the agent can continuously adapt its actions based on the uncertainty in the environment.
- **Critic Networks**: SAC employs two critic networks, $Q_1(s,a|\theta^{Q_1})$ and $(Q_2(s,a|\theta^{Q_2}))$, to mitigate overestimation bias—a common problem in Q-learning. The critics' role is to assess each action's value and compute the expected return. The target Q-value in SAC is given by:

$$y = r + \gamma\left(\min_{i=1,2} Q_i\,(s', a') - \alpha \log \pi\,(a'|s')\right)$$

where α scales the entropy term, and $\pi(a'|s')$ is the policy's entropy at the next state-action pair. This dual-critic mechanism reduces the likelihood of overestimating the potential value of future states, stabilizing learning.

SAC's focus on both exploration and reliable value estimation makes it highly effective for manipulation tasks with uncertain outcomes.

5.1.3 Twin Delayed Deep Deterministic Policy Gradient (TD3)

TD3 is a refinement of DDPG, designed to counter overestimation bias through several key mechanisms, such as twin critics and delayed actor updates.

- **Twin Critic Networks** : Like SAC, TD3 uses two critic networks, denoted as $Q_1(s,a|\theta^{Q_1})$ and $(Q_2(s,a|\theta^{Q_2}))$, to reduce overestimation in Q-value estimates. When calculating the target Q-value, TD3 selects the minimum value from the two critics, ensuring a conservative estimate:

$$y = r + \gamma \min_{i=1,2} Q_i\left(s', \mu(s'|\theta^{\mu})\right)$$

- **Actor Network** : The actor $\mu(s|\theta^{\mu})$ outputs a deterministic action based on the current policy. However, unlike DDPG, TD3 updates the actor less frequently than the critic (e.g., every two critic updates). This helps the critic stabilize before the actor makes adjustments, improving convergence. By combining these strategies, TD3 reduces the likelihood of overestimation errors and enhances policy stability, which is crucial for continuous-action, high-stakes tasks like robotic manipulation.

5.2 Hindsight Experience Replay (HER)

HER is employed with off-policy methods to improve sample efficiency, especially in environments with sparse rewards. When an agent fails to reach the target, HER modifies the goal to align with the agent's actual achieved state, generating synthetic positive feedback from previously unsuccessful attempts. This approach effectively turns failed experiences into useful learning opportunities, accelerating training for complex tasks.

5.3 On-Policy Methods

In contrast, on-policy methods train on data generated by the current policy only, leading to more stable learning. They are particularly useful for scenarios requiring incremental improvements without drastic policy shifts.

5.3.1 Proximal Policy Optimization (PPO)

PPO improves the stability of on-policy learning by limiting policy updates. It uses a clipped surrogate objective to prevent overly large updates, ensuring the policy does not stray far from the current one.

- **Actor Network :** The actor in PPO learns to maximize the probability of selecting actions that yield high returns. Unlike off-policy methods, PPO maintains exploration within a constrained parameter space. The actor's updates are controlled by the following objective:

$$J^{\text{PPO}}(\theta) = E\left[\min\left(r(\theta)\widehat{A_t}, \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)\widehat{A_t}\right)\right]$$

  where $(r(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)})$ represents the ratio of new to old policy probabilities, $\widehat{A_t}$ is the advantage estimate, and ϵ is a clipping threshold. This objective balances exploration with policy stability.
- **Critic Network :** The critic network estimates the expected cumulative reward for each state, providing a baseline against which the actor's chosen actions are compared. The critic helps reduce variance in the actor updates, leading to smoother policy improvement.

By ensuring gradual, consistent updates, PPO is particularly suited for tasks requiring reliable convergence, like robotic manipulation.

5.3.2 Advantage Actor-Critic (A2C)

A2C synchronizes multiple agent experiences to calculate a more accurate advantage estimate, reducing variance in policy updates and improving performance in sparse-reward environments.

- **Actor Component :** The actor in A2C selects actions based on the advantage function, which measures the value of a particular action relative to other potential actions. The advantage function $\widehat{A_t}$ is defined as:

$$\widehat{A_t} = R_t - V(s_t|\theta^V)$$

  where R_t represents the total return and $(V(s\_t|\theta^V)$ is the state value. By encouraging actions that perform better than average, the actor gradually refines the policy.

- **Critic Component :** The critic evaluates the baseline value for each state, serving as a comparison for the actor's chosen actions. This reduces the variance in the policy updates, providing a stable learning signal.

A2C's synchronous learning setup and advantage-based updates make it effective for real-time applications, balancing exploration with consistent policy improvement.

5.4 Generalized Advantage Estimation (GAE)

GAE helps balance the trade-off between bias and variance in advantage estimation by adjusting the weight of long-term versus immediate rewards. GAE is crucial for on-policy methods, as it provides smoother and more stable advantage estimates.

The GAE advantage calculation is given by :

$$A_t^{\widehat{GAE(\gamma,\lambda)}} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}$$

where δ_t is the TD error, γ the discount factor, and λ a weighting parameter. By controlling λ, we can adjust the trade-off between short-term and long-term rewards, yielding more robust learning.

5.5 Hyperparameters

For optimal performance across tasks, we tuned the following key hyperparameters:

- Learning Rate : Set to 0.001 for off-policy methods and 0.0003 for on-policy methods, balancing learning speed with stability.
- Discount Factor (γ) : Fixed at 0.99 to prioritize long-term reward accumulation.
- Clipping/Entropy Regularization : Used in PPO (clipping threshold of 0.2) and SAC (entropy coefficient α to manage policy updates and promote exploration.

6. Experiments and Results

This section details the experimental setup and presents results for the three robotic manipulation tasks: reaching, pushing, and pick-and-place. Frameworks such as [17] provide standardized setups for benchmarking robotic manipulation tasks, allowing fair comparisons between RL algorithms in diverse scenarios. Each task's outcomes are analysed with a focus on quantitative metrics, including success rate and convergence. Clear visualizations are provided to support the performance insights.

6.1 Experiment Setup

6.1.1 Environment Configuration :

The three tasks required varied manipulation skills: reaching involved precise arm positioning, pushing entailed controlled force application, and pick-and-place combined grasping and lifting actions. Each task's environment challenged the agent's capabilities across different skill-sets fundamental to robotic manipulation.

The training episode count was set according to task complexity, with reaching trained for 1000 episodes, while pushing and pick-and-place were each trained for 30,000 episodes.

6.1.2 Evaluation Metrics :

- Success Rate : Defined as the proportion of episodes where the agent successfully completed the task within the episode's time limit.
- Average Reward : Calculated as the cumulative reward averaged over episodes.
- Convergence Analysis : Stability in learning curves was observed to assess each algorithm's effectiveness in adapting to the task environment.

6.2 Results

The results for each task are organized below, focusing on performance differences across off-policy (DDPG, SAC, TD3) and on-policy (PPO, A2C) algorithms.

6.2.1 Reaching Task

The reaching task tested the agent's precision in moving to a target location, serving as the simplest task in this experiment.

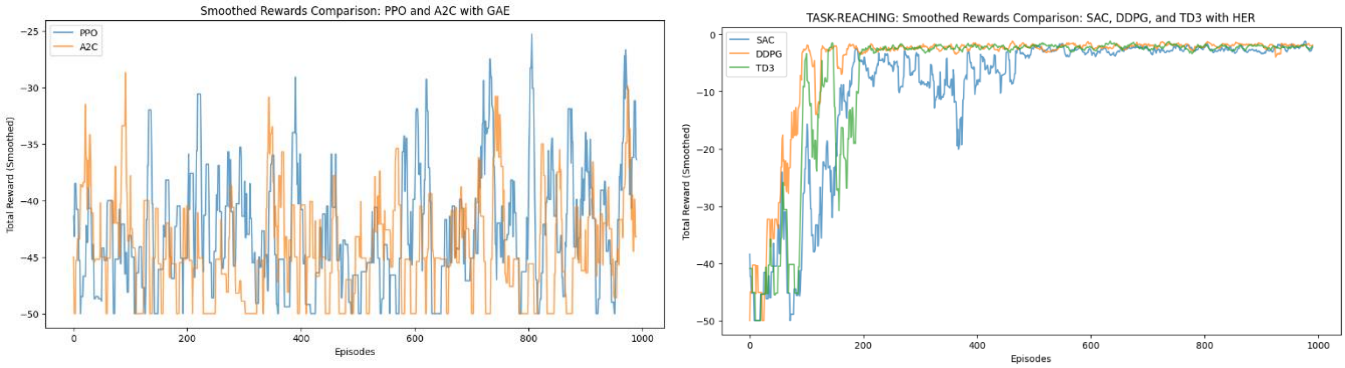


Figure 6.2.1 : Rewards plotted and compared for 1000 episodes

Performance :

1.) Off-Policy :
   - DDPG excelled by quickly reaching a success rate of 0.99-1.0 within 200 episodes, proving effective for precise control in simpler tasks.
   - TD3 followed closely, although its twin-critic structure took slightly longer to achieve stable convergence.
   - SAC displayed slower convergence due to entropy-based exploration, eventually reaching high success rates.

2.) On-Policy :
   - PPO attained a maximum success rate around 0.45 but failed to fully converge, reflecting inherent difficulties in handling continuous control tasks.
   - A2C was more variable, with lower success rates (peaking at 0.33) and less consistent performance.

6.2.2 Pushing Task

The pushing task involved dynamic interactions, where the agent applied force to move an object to a target position adding a layer of complexity.

,

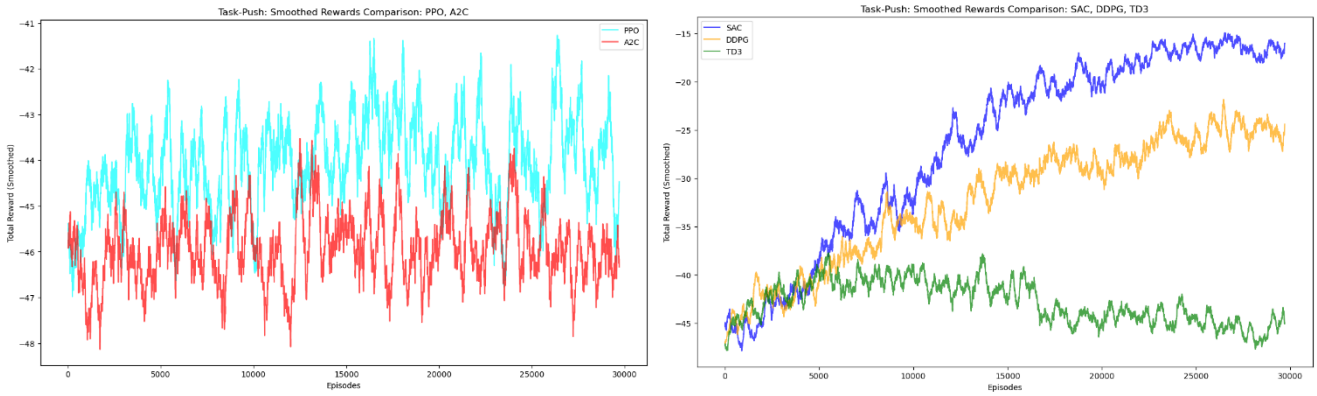


Figure 6.2.2 : Rewards plotted and compared for 30000 episodes

Performance :

1) Off-Policy :
   - SAC achieved the best performance with a success rate of 0.95 by the end of training, benefiting from entropy-based exploration in navigating dynamic object interactions.
   - DDPG reached a peak success rate of 0.73, with limitations in adapting to the contact-based interactions in pushing.
   - TD3 peaked at a success rate of 0.45 around 15,000 episodes, then dropped to around 0.2 by the end of training, indicating a decline in adaptive performance.
2) On-Policy :
   - PPO outperformed A2C, achieving a maximum success rate around 0.25, though both algorithms struggled with stability.

6.2.3 Pick-and-Place Task

The pick-and-place task, which required sequential actions of grasping, lifting, and positioning, was the most complex task and tested the algorithms' ability to coordinate multiple actions precisely.

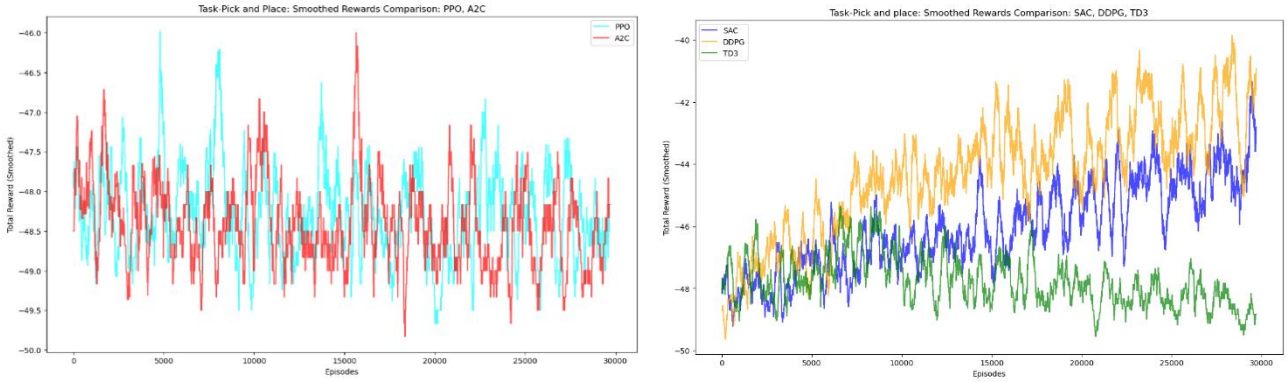


Figure 6.2.3 : Rewards plotted and compared for 30000 episodes

Performance :

1) Off-Policy :
   - DDPG achieved the highest success rate at 0.25, a notable result given the task complexity and limited episodes.
   - SAC followed(0.19), though its exploratory approach posed challenges for sequential actions.
   - TD3 showed limited success(0.12), struggling with task stages requiring precise control.
2) On-Policy :
   A2C achieved a peak success rate around 0.11 but struggled with the multi-phase task complexity.

7. Discussion

Task/Algorithm	DDPG	SAC	TD3	PPO	A2C
Reach	1.00	1.00	1.00	0.45	0.33
Push	0.73	0.95	0.45	0.25	0.16
Pick and Place	0.25	0.19	0.12	0.10	0.11

Table 7.1 : Maximum success rates of each algorithm for every task

7.1 Off-Policy Methods: Successes and Limitations

In this project, off-policy methods demonstrated varying success across tasks. DDPG, which performed exceptionally well for both reaching and pick-and-place tasks, showed robust convergence and stability in these scenarios. Multi-goal RL has been explored extensively in [18], demonstrating its utility in handling sparse-reward robotic manipulation tasks. This aligns closely with the benefits observed from HER in our off-policy experiments. The importance of HER in addressing sparse reward challenges is further validated by [24], which demonstrated significant performance improvements in multi-goal RL environments. For reaching, DDPG quickly achieved a success rate near 1.0 within 200 episodes, likely due to its continuous action space handling and efficient policy updates using deterministic gradients, which are advantageous in precise positioning tasks. Similarly, DDPG's performance in pick-and-place tasks suggests that its capability to learn stable and efficient control policies aligns well with the complexity of gripping and moving objects, where precision is key. However, for the pushing task, DDPG's performance was less successful. This may be due to the unpredictable physics involved in pushing, which introduces dynamics that DDPG's deterministic approach doesn't capture as effectively.

SAC excelled in the pushing task, reaching success rates of 0.93-0.95. Its stochastic nature in policy exploration, combined with the entropy regularization, likely allows SAC to handle the variability and uncertainty involved in the pushing task better than DDPG or TD3. SAC's exploration helps adapt to unpredictable forces that arise when pushing an object in the environment. However, SAC's results in reaching and pick-and-place did not surpass those of DDPG, suggesting that SAC's exploration-heavy approach may add unnecessary variability in tasks requiring precise, directed actions.

TD3, despite being very effective in reaching tasks with a high success rate, struggled with pushing and pick-and-place. While TD3 incorporates twin critics to reduce overestimation bias, which helps in stable learning, its performance deteriorated over time in pushing and pick-and-place, indicating instability. This may be attributed to TD3's sensitivity to the noise scale used in exploration. Recent advancements in actor-critic methods, as discussed in [23], address overestimation bias, a common issue in RL algorithms, particularly TD3, which has proven effective in robotic manipulation tasks. The agent's performance dropping in pushing, particularly after 15,000 episodes, suggests that noise-induced exploration may lead it away from optimal policies in environments requiring sustained, goal-oriented actions like pushing and object manipulation.

7.2 On-Policy Methods: Limitations and Non-Convergence

On-policy methods, PPO and A2C, struggled with all tasks, failing to converge to a reliable success rate and plateauing at lower success levels (maximum of around 0.45 for reaching, 0.25 for pushing, and 0.11 for pick-and-place). This outcome reflects a fundamental limitation of on-policy methods in complex robotic tasks requiring extensive exploration and fine-tuned adjustments. Since on-policy approaches only update from the most recent policy and data, they often lack the ability to retain beneficial past experiences, which limits their effectiveness in continuous, high-dimensional tasks. Extensions to trust-region methods in RL, such as those described in [21], aim to address instability issues in deep RL, particularly for large-scale robotic environments.

GAE helped stabilize the learning to an extent, particularly in PPO, but on-policy approaches continued to show limited convergence. In environments with high variability, such as pushing, the limited exploration of on-policy methods makes it difficult for the agent to adapt to unpredictable dynamics, leading to suboptimal policies. This indicates that on-policy methods are less suited to tasks requiring broad exploration or where replay of past experiences is advantageous.

8. Conclusion and Future Works

This project has provided an in-depth evaluation of various reinforcement learning (RL) algorithms applied to robotic manipulation tasks in a simulated environment using the PyBullet framework [15]. Our findings suggest that off-policy methods, specifically DDPG, SAC, and TD3, consistently outperform on-policy methods such as PPO and A2C in achieving higher success rates for tasks like reaching, pushing, and pick-and-place [16][17][18]. Among the off-policy methods, DDPG excelled in both reaching and pick-and-place tasks due to its stable learning approach [16][22], while SAC proved more effective in handling the complexities of pushing, owing to its exploration-driven policy [18][22]. However, TD3 showed instability over extended episodes in pushing and pick-and-place, suggesting sensitivity to task complexity and noise settings [23].

If these algorithms were trained for even more episodes, it is likely that their performance would continue to improve, particularly for tasks that require precision and stability. However, increasing the number of training episodes alone may not address all limitations. For future work, exploring advanced off-policy methods such as Soft Actor-Critic with Automatic Entropy Tuning (SAC-AE) [22] or Twin Delayed Deep Deterministic Policy Gradient with Prioritized Experience Replay (TD3-PER) [23] could enhance stability and convergence speed. Additionally, hybrid approaches that combine the advantages of off-policy exploration with structured on-policy learning could offer new avenues for improvement [21]. Recent studies such as [25] propose scalable off-policy RL methods like Advantage-Weighted Regression, focusing on improving sample efficiency and policy robustness in complex robotic tasks.

Another key direction is the integration of the best-performing algorithms into physical robotic systems. Transferring policies learned in simulation to physical robots would enable safe, adaptable manipulation in real-world environments [15][19]. This could be achieved through sim-to-real transfer techniques like domain randomization and fine-tuning with real-world data [15]. Combining this approach with robust off-policy algorithms could facilitate the application of robotic arms in diverse fields such as warehouse automation, healthcare, and manufacturing. Improved simulation fidelity and expanded datasets could further enhance the adaptability and precision of RL policies in real-world scenarios, helping bridge the gap between simulated training and physical deployment [14][20].

9. References

[1] T.P. Lillicrap, et al., “Continuous control with deep reinforcement learning,” ICLR, 2016.

[2] T. Haarnoja, et al., “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” ICML, 2018.

[3] S. Fujimoto, et al., “Addressing function approximation error in actor-critic methods,” ICML, 2018.

[4] M. Andrychowicz, et al., “Hindsight experience replay,”z NeurIPS, 2017.

[5] J. Schulman, et al., “Proximal policy optimization algorithms,” arXiv preprint arXiv:1707.06347, 2017.

[6] V. Mnih, et al., “Asynchronous methods for deep reinforcement learning,” ICML, 2016.

[7] D. Kalashnikov, et al., “QT-Opt: Scalable deep reinforcement learning for vision-based robotic manipulation,” CoRL, 2018.

[8] M. Horgan, et al., “Distributed prioritized experience replay,” arXiv preprint arXiv:1803.00933, 2018.

[9] T. Schaul, et al., “Prioritized experience replay,” ICLR, 2016.

[10] OpenAI, et al., “Learning dexterous in-hand manipulation,” arXiv preprint arXiv:1808.00177, 2018.

[11] Y. Duan, et al., “RL2: Fast reinforcement learning via slow reinforcement learning,” arXiv preprint arXiv:1611.02779, 2016.

[12] Levine, S. et al. (2016). 'End-to-end training of deep visuomotor policies.' Journal of Machine Learning Research.

[13] Gu, S. et al. (2017). 'Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates.' Robotics: Science and Systems (RSS).

[14] Tan, J. et al. (2018). 'Sim-to-Real: Learning Agile Locomotion for Quadruped Robots.' Robotics: Science and Systems (RSS).

[15] Coumans, E. and Bai, Y. (2016). 'PyBullet: Physics simulation for games, visual effects, robotics, and machine learning.'

[16] Henderson, P. et al. (2018). 'Deep reinforcement learning that matters.' Proceedings of the AAAI Conference on Artificial Intelligence.

[17] Cabi, S. et al. (2019). 'Framework for Benchmarking RL Algorithms in Robotic Manipulation.' CoRL.

[18] Plappert, M. et al. (2018). 'Multi-goal reinforcement learning: Challenging robotics environments and request for research.' CoRL.

[19] Pong, V. et al. (2018). 'Temporal Difference Models: Model-Free Deep RL for Model-Based Control.' CoRL.

[20] Dhariwal, P. et al. (2017). 'Proximal Policy Optimization Algorithms.' arXiv preprint arXiv:1707.06347.

[21] Wu, Y. et al. (2017). 'Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation.' NIPS.

[22] Haarnoja, T. et al. (2018). 'Reinforcement Learning with Deep Energy-Based Policies.' CoRL.

[23] Fujimoto, S. et al. (2018). 'Addressing Function Approximation Error in Actor-Critic Methods.' ICML.

[24] Andrychowicz, M. et al. (2017). 'Hindsight Experience Replay.' NIPS.

[25] Kostrikov, I. et al. (2018). 'Advantage-Weighted Regression: Simple and Scalable Off-Policy RL.' arXiv preprint.