Benchmarking Model-Based and Model-Free Reinforcement Learning for Robotic Manipulation: DreamerV3 versus SAC

Tushar Deshpande
Virginia Tech
tushard@vt.edu

Abstract

In robotic control tasks where collecting data is considered a costly adventure, model-based reinforcement learning (MBRL) offers a great potential for much higher sample efficiency compared to traditional model-free reinforcement learning methods. In this project, we intend to explore how well DreamerV3, a leading MBRL algorithm performs on the FetchReach-v3 task, which is a challenging benchmark for robotic manipulation. We remodel DreamerV3 to work seamlessly with a custom FetchReach wrapper built for the Gymnasium framework and compare its performance against the widely used model-free Soft Actor-Critic (SAC) algorithm.

With our experiment, we reveal key trade-offs between these two approaches in terms of training speed, learning stability, and overall performance. As per our results, DreamerV3 shows a clear advantage in terms of sample efficiency during the early phases of training since it shows better rewards scales. However, the SAC tends to achieve better long-term performance when dense rewards are available. We also show the difficulties that are presented by the FetchReach environment, such as sparse rewards and the difficulty of accurately modeling environment dynamics. Overall, our benchmark results shows the practical advantages and disadvantages of using model-based versus model-free methods in goal-directed robotic tasks.

1. Introduction

In applications such as robotics, autonomous systems and industrial control, using sequential decision making processes is a powerful technique. Reinforcement learning (RL) is one such example of sequential decision processes and it has shown great promise. Model-free RL algorithms like Soft Actor-Critic (SAC) [1], Deep Q-Networks (DQN) [2] and Proximal Policy Optimization (PPO) [3] have shown strong results across many benchmark environments. However, a major limitation of these methods is their low sample efficiency. These algorithms often require millions of interactions with the environment to learn effective policies. This makes them impractical for real-world robotics tasks where data collection is considered both time-consuming and expensive. On the other hand, Model-based reinforcement learning (MBRL) offers a positive alternative by learning a model of the environment's dynamics. The algorithm works by using the dynamics model to simulate future trajectories and update its policy more efficiently, instead of relying solely on real interactions. The earlier MBRL approaches that were implemented [4] and [5], were prone to instability and accumulated errors over time, however recent progress such as learning in the latent spaces, modeling stochastic dynamics and using imagined rollouts have emerged as renewed interest in the field. DreamerV3 [6] stands out among these newer methods which was published in Nature in 2024. This algorithm uses a latent world model alongside actor-critic training to deliver impressive performance on complex control tasks with significantly fewer real interactions. Building on these advances, the DayDreamer framework [7] extends world model learning to physical robots. They demonstrate successful real-world deployment of Dreamer-based agents without environment-specific engineering.

While DreamerV3 has shown impressive performance on the widely used benchmarks like the DeepMind Control Suite [8] and OpenAI Gym, its effectiveness and efficiency on more realistic, robotics oriented tasks has not been thoroughly examined. In this study, we intend to address this gap by evaluating the DreamerV3 algorithm on the FetchReach-v3 environment, which is a task from the Gymnasium Robotics benchmark suite. Although its relatively simple in structure, FetchReach poses meaningful challenges. It is goal-conditioned task whcih offers sparse rewards and demands precise and coordinated motor control. Its physics based kinematics also make it more representative of real-world robotic manipulation. This makes it a suitable environment to compare the strengths and weaknesses of model-based and model-free reinforcement learning. To carry out this comparison, we have devel-

oped a Gymnasium wrapper which is compatible with the FetchReach-v3 environment and trained agents using both the DreamerV3 and Soft Actor-Critic (SAC) algorithms. We initially selected SAC as a robust model-free baseline known for its training stability and effective entropy-driven exploration. These qualities are known to be particularly useful in sparse reward scenarios. DreamerV3, in contrast, uses a model-based approach by learning a latent representation of the environment and training its policy on simulated trajectories which are called the imagined rollouts. This approach offers a fundamentally different way of learning from data which was generated by the model itself. Lastly with this setup, our final goal is to explore key questions about the performance, efficiency and dynamics of these two learning paradigms in the context of the robot control tasks. Hence for this we aim to answer the following questions:

- How does DreamerV3 perform in terms of sample efficiency when compared to SAC on a robotic control task with sparse rewards?
- Can latent-space imagination which is used in Dreamer compensate for model inaccuracies that arise in highly dynamic environments like FetchReach?
- What are the failure modes of each method in terms of generalization, stability and reward convergence?

Our initial experiments on the half cheetah and reacher environments, suggest that DreamerV3 can exhibit fast early learning. They require significantly fewer interactions than the SAC algorithm to begin improving. This behaviour reflects the main advantage of model-based reinforcement learning which is greater sample efficiency. However, as training progresses, particularly in environments with dense rewards we have found that SAC eventually surpasses DreamerV3 in final performance. This supports the current existing observations as well that model-free methods which are slower to learn, often tend to be more robust over time. They benefit from direct optimization of the return without relying on potentially imperfect dynamics models.

We also investigate the training instability observed with DreamerV3, which appears to result from the world model's difficulty in accurately capturing abrupt positional changes. This often leads to poor imagined rollouts and degraded policy learning in the end. To address these challenges, we explore reward shaping and entropy annealing as potential strategies for improving training stability across both model-based and model-free approaches.

Our key contributions are as follows:

- We benchmark DreamerV3 and SAC on the FetchReach-v3 environment using a custom built wrapper for the Gymnasium interface.
- We provide an in-depth comparison of model-based and model-free RL approaches under sparse and shaped reward settings.
- We provide practical insights into the trade-offs involved in applying model-based reinforcement learning to robotic manipulation tasks, focusing mianly on metrics like convergence and the accuracy of environment modeling

By highlighting the performance metrics in the DreamerV3 environment which comes under the robotic domain, this study makes an effort to contribute to the growing body of literature on deploying world models in practical robotic control tasks.

2. Methods

In this section, we describe the methodology used to evaluate and compare DreamerV3 and Soft Actor-Critic (SAC) algorithms on the FetchReach-v3 task. We start by outlining the setup of the FetchReach-v3 environment, then provide an in-depth overview of the two reinforcement learning algorithms. Finally, we mention the training procedures and evaluation metrics along with the key implementation details relevant to this benchmark study.

2.1. Environment: FetchReach-v3

The FetchReach-v3 environment is part of the Gymnasium Robotics suite. This environment provides realistic robot simulation tasks using the MuJoCo physics engine. The agent controls a 7-DoF Fetch robotic arm and must move its end-effector to reach a randomly sampled 3D target position. Each episode begins with a new target and ends when either of the two scenarios take place: 1) the arm reaches the goal within a fixed tolerance 2) the maximum time steps have been reached for that episode.

The observation space for this environment includes:

- The current position of the gripper (3D vector).
- The relative position between gripper and goal.
- Velocities and control inputs for the joints

The agent operates in a continuous action space, controlling three-dimensional Cartesian movements (dx, dy, dz), along with a gripper control dimension that remains unused in the FetchReach task. By default, the environment uses a sparse reward: the agent receives a reward of 0 only when the gripper reaches within 0.05 units of the target, and -1 otherwise. To provide a more informative learning signal, we also experiment with a shaped reward version, where the agent is penalized based on the negative Euclidean distance to the goal.

2.2. Soft Actor-Critic (SAC)

We use the Soft Actor-Critic (SAC) algorithm [1], a widely-used model-free and off-policy reinforcement learning method which makes use of the maximum entropy framework. The SAC algorithm aims to learn a stochastic policy which simultaneously maximizes the expected cumulative reward and policy entropy, thereby promoting effective exploration in high-dimensional continuous action spaces.

In the SAC algorithm, the agent consists of the following components:

- **Actor network:**: This network policy is represented as a diagonal Gaussian distribution, where the neural network predicts the mean and log standard deviation of the action distribution. To enable efficient backpropagation via stochastic actions, actions are sampled using the reparameterization trick.
- **Critic networks:** To reduce the overestimation bias, the SAC algorithm uses two separate Q-function approximators, by following the Clipped Double Q-learning approach. Both the critic networks are trained by minimizing the Bellman error relative to target Q-values.
- **Entropy coefficient (α):** The SAC algorithm uses a temperature parameter to balance reward maximization with exploration. This parameter can either be tuned manually or adjusted automatically by optimizing a temperature loss parameter, which helps to drive the policy entropy towards a predefined target value.

We use the Gymnasium-compatible SAC implementation from Stable-Baselines3 [10], with a customized environment wrapper that normalizes both actions and observations. The policy and value networks are composed of two hidden layers each with 256 units. We have used the ReLU activations for our implementations here. SAC is trained for 1 million time steps per run, with updates performed at every step using a replay buffer of size one million (10^6).

2.3. DreamerV3

DreamerV3 is a latent imagination-based model-based reinforcement learning (MBRL) algorithm that separates the process of dynamics modeling from the policy learning part. Instead of directly using environment transitions to update the policy, DreamerV3 first learns a small latent world model from real interactions with the environment. This model is then used to generate the imagined trajectories in the latent space. This enables the agent to learn and improve its policy efficiently without further interacting with the real environment during each update cycle, thus make it more sample efficient.

Below are the core components of the Dreamer V3 algorithm:

- **World Model:** A recurrent stochastic state-space model (RSSM) which consists of a transition model, an observation decoder, a reward predictor and an optional discount predictor.
- **Latent Policy:** The actor and critic are trained fully in the latent space by using trajectories generated from the world model through the imagination process.
- **Free-Form Dynamics:** DreamerV3 introduces a learned slow feature representation which enhances the generalization across continuous control tasks. This eliminates the need for the gradient balancing techniques which was required in the earlier versions.

For our implementation, we use the open-source PyTorch implementation of DreamerV3 adapted from official dreamer GitHub repo [9], later modifying it for the FetchReach-v3 environment. Specifically, the details of our implementation are as below:

- We adapt DreamerV3 to work with low-dimensional state inputs derived from the robot's internal sensors. We use the agent's current state and the desired goal provided by the environment, instead of pixel-based observations.
- To create a goal-aware input, we concatenate the observation with the goal vector and pass it through a feedforward encoder to produce latent features for the world model.
- The world model is composed of an encoder, a GRU-based transition model and a reward predictor. These components are trained on real transitions collected from the environment and stored in a replay buffer augmented with Hindsight Experience Replay (HER).
- HER is implemented by relabeling the desired goal with a goal actually achieved later in the trajectory, and updating the corresponding reward. This helps the agent to learn more effectively from sparse reward signals.
- Both the actor and critic operate in the latent space, using imagined trajectories generated by the world model to update the policy and value functions.
- During imagination, the agent performs roll outs of the world model for a fixed number of steps, starting from the latent representation of a real environment state in order to simulate future outcomes.

- The agent's actions are continuous and scaled to comply with MuJoCo's actuator limits, ensuring stable and realistic control within the simulator.

- We monitor the training progress using TensorBoard, where metrics such as episode reward and success rate are logged for later diagnostic purposes.

2.4. Training and Evaluation Protocol

To ensure a fair comparison between model-free and model-based approaches, we apply a consistent training and evaluation procedure across both the SAC+HER and DreamerV3 agents:

- **Episode length:** Each episode is limited to 50 time steps, following the default configuration of `FetchReach-v3`.

- **Evaluation:** Agents are evaluated every 5,000 environment steps over 10 episodes using deterministic actions. A success is recorded when the goal is reached within a 0.05-unit distance threshold.

- **Training budget:** SAC is trained for up to 10,00,000 environment steps, while DreamerV3 is trained for 100,000 steps.

- **Random seeds:** To account for variability, each agent is trained using 5 different random seeds. We report the mean and standard deviation of the final success rate and sample efficiency.

- **Reward shaping:** In dense reward experiments, the environment is modified to return the negative Euclidean distance between the gripper position and the goal, i.e., $r_t = -\|g - x_t\|_2$. This reward signal is used consistently for both agents during training.

- **HER relabeling:** To address sparse rewards, both agents make use of the Hindsight Experience Replay (HER). For SAC, we use the built-in HER module from Stable-Baselines3 with `future` goal sampling. For DreamerV3, we implement a custom HER mechanism that relabels transitions using future achieved goals within each episode.

- **Logging:** All training runs are logged using TensorBoard. For SAC, we track episode rewards, success rates, and loss metrics through the Stable-Baselines3 logger. For DreamerV3, we log episodic returns, success rates, and losses for the world model, actor and critic components.

2.5. Implementation

All experiments are conducted on the ARC HPC cluster at Virginia Tech. Each agent runs on a single NVIDIA A100 GPU or Tesla T4 GPUs with 80GB/48GB memory respectively. For DreamerV3, the training consumed roughly 6GB of GPU memory while SAC uses less than 2GB of GPU power.

3. Experimental Results

3.1. Soft Actor Critic

We evaluate the fine-tuned SAC + HER agent on the `FetchReach-v3` task using a dense reward formulation. The agent is trained for 1 million environment steps, with evaluation and checkpointing performed every 5,000 steps. Key learning metrics are recorded via TensorBoard and summarized below.

3.1.1 Training Dynamics

Figure [1] presents the loss curves for the actor, critic and the entropy coefficient modules. The **actor loss** quickly stabilizes around a narrow band near 1.0, indicating that the policy has converged to a consistent action distribution. The **critic loss** shows a steep initial decline within the first 100k steps, followed by low and stable values throughout the remainder of training, suggesting effective value estimation.

The **entropy coefficient loss** remains centered around zero with moderate fluctuations, consistent with an auto-tuned entropy strategy that balances exploration and exploitation adaptively.

3.1.2 Reward and Success Rate Trends

Figure [2] shows the evolution of the mean episode reward and the goal success rate across training. While the **mean reward** improves steadily during early training, it exhibits high variance later on, suggesting the agent occasionally fails to generalize across diverse goal configurations. The **success rate** fluctuates between 20% and 60% for most of training, indicating that although the agent learns to reach goals reliably in some scenarios, performance remains inconsistent overall.

3.1.3 Observations

Despite stable convergence in value estimation and policy learning, the success rate touches the 60% mark only sometimes. This suggests that the shaped reward alone may not be sufficient to drive high final performance without additional mechanisms such as curriculum learning, increased HER relabeling frequency, or longer training duration. However, the results demonstrate that the fine-tuned

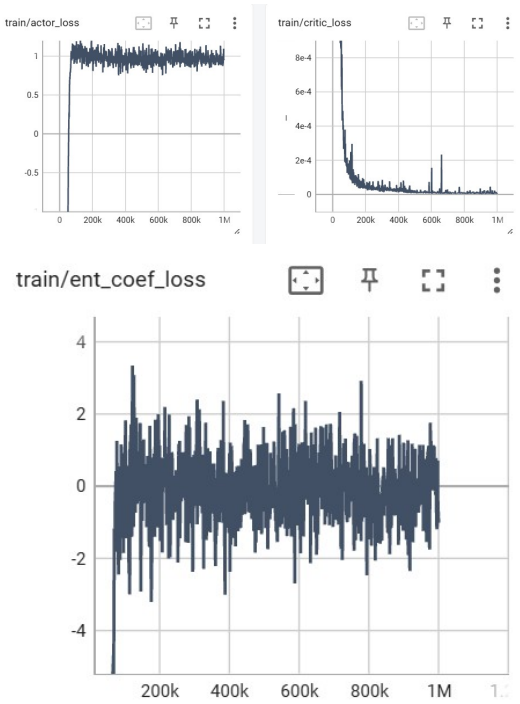

Figure 1. Training loss curves for actor, critic and entropy coefficient modules.

SAC agent is able to leverage dense rewards and HER to achieve moderate success on this challenging continuous control task.

3.2. Dreamer

We evaluate the DreamerV3 agent on the `FetchReach-v3` environment using a dense reward formulation and hindsight relabeling. The agent is trained for 100,000 environment steps using low-dimensional state and goal features as input. Throughout training, we log essential metrics to monitor convergence trends and evaluate policy performance.

3.2.1 Training Dynamics

Figure [3] shows the training loss curves for the actor, critic, and world model components. The actor loss shows a clear decreasing trend, which indicates that the policy is progressively optimizing toward higher expected returns in the imagined latent space. The critic loss is seen to be low throughout training and exhibits diminishing variance, suggesting stable value estimation across imagined rollouts. Interestingly, the world model loss also decreases rapidly within the first 20k steps and converges to a low value, which indicates that the RSSM is able to effectively capture environment dynamics from the goal-conditioned transitions.

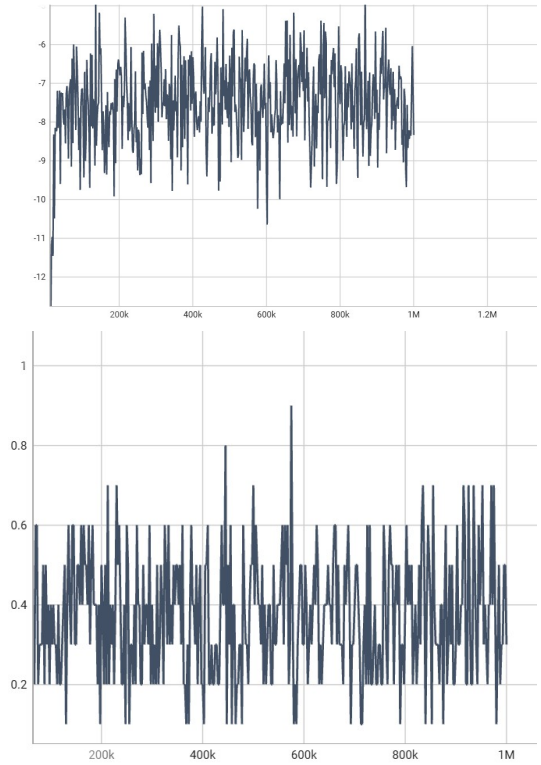

Figure 2. Above: Mean episode reward; Down: Success rate evaluated every 5k steps.

3.2.2 Reward Trends

Figure [4] presents the episode-level reward across training. While the **reward signal** remains noisy due to the sparse nature of successful goal completions, the range narrows over time, and extreme negative values become less frequent. This indicates gradual improvement in the agent's ability to reach goals, albeit with notable variance.

3.2.3 Observations

The training curves suggest that the DreamerV3 agent is able to learn meaningful representations of environment dynamics and optimize a goal-conditioned policy in latent space. While rewards remain highly variable and low due to task difficulty and stochastic rollouts, the agent exhibits steady improvement without collapse. The fast convergence of the world model highlights the sample efficiency of Dreamer-style agents, which can perform multiple policy updates per environment step.

The success results however suggest that while DreamerV3 demonstrates stable learning dynamics and effective world model training, it struggles to achieve consistent goal-reaching behavior on FetchReach-v3 in its current form. The lack of successful completions shows that further tuning — such as improved exploration strategies

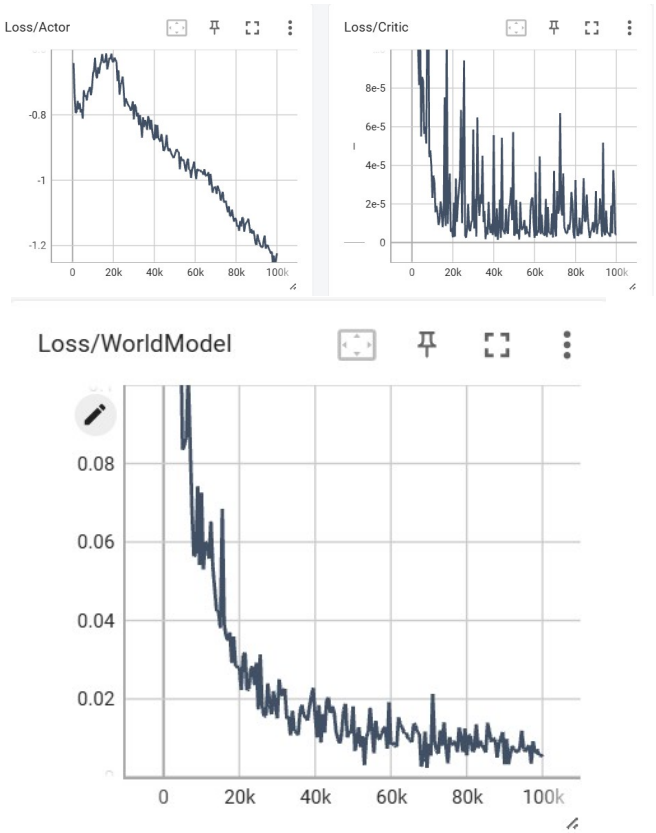

Figure 3. Training loss curves for actor, critic and world model components for DreamerV3.

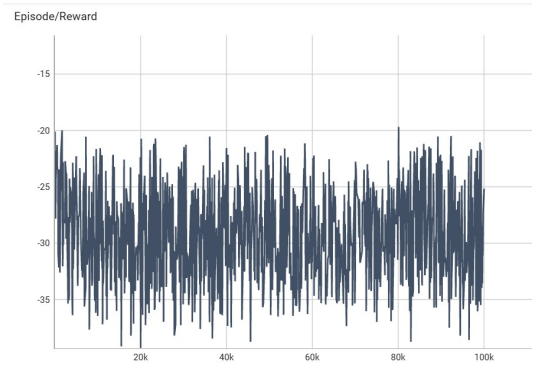

Figure 4. Per-episode reward over time for DreamerV3 on FetchReach-v3.

or curriculum learning may be necessary to solve this low-dimensional robotic task.

4. Discussion

Here we examine and compare the performance of the SAC+HER and DreamerV3 agents on the FetchReach-v3 environment, drawing conclusions on both the training metrics and qualitative observations.

4.1. Soft Actor Critic: Strengths and Limitations

The Soft Actor-Critic algorithm, combined with Hindsight Experience Replay (HER), demonstrated competitive learning under the dense reward settings. The training losses for both actor and critic networks stabilized relatively early and the entropy coefficient adapted effectively to maintain a good balance between exploration and exploitation. As training progressed, average rewards increased, and the agent was able to achieve success rates greater than 60% and even touched 80% which shows that the goal was been able to meet.

However, the agent's performance showed noticeable fluctuations within the episode steps. This instability may stem from the limited expressiveness of the dense reward function or insufficient frequency and diversity in HER relabeling. Since SAC is entirely model-free, it may also struggle to generalize across a wide distribution of goal states. While its value estimates are generally stable, getting more robust performance may require extended training even beyond the 1M steps, different goal selection strategies or a structured learning curriculum to guide progress.

Nevertheless, the SAC algorithm performs much better than the Dreamer V3 on the Fetch task with minimal fine tuning done. It does take more to train on than the Dreamer algorithm but we can see better success rates when compared with the later.

4.2. DreamerV3: Strengths and Limitations

DreamerV3 excelled in learning compact representations of the environment, as reflected in the smooth and fast convergence of the world model loss. Actor and critic networks showed consistent learning signals, suggesting effective policy optimization in latent space. These trends highlight DreamerV3's strength in sample efficiency, as it leverages imagined rollouts to perform many policy updates from relatively few real interactions.

However, this internal progress did not translate into strong task performance. The agent's success rate remained close to zero, and episode rewards stayed low and stagnant throughout the training period. A likely cause can be the short horizon of FetchReach-v3, which may limit the effectiveness of imagination-based planning. The sparse reward structure, even with the use of HER adds another layer of difficulty. It limits meaningful feedback for learning value estimates in the latent space.

An additional challenge arises from discrepancies between real environment transitions and the imagined rollouts generated by the learned dynamics model. While imagined trajectories are central to Dreamer's efficiency, they may lack sufficient variability or goal-specific information in the short-horizon settings which can badly affect the performance. This can cause the policy to rely on inaccurate latent predictions, ultimately affecting the goal-reaching per-

formance. Furthermore, HER in our implementation modifies only real trajectories, not imagined ones, which can limit the propagation of successful experiences into the imagination space. Extending HER to apply within latent rollouts or introducing goal conditioning during imagination can help improve credit assignment and planning accuracy.

4.3. Comparison and Future Directions

SAC+HER showed gradual but tangible progress toward task completion, though with some inconsistency. DreamerV3, in contrast, exhibited strong internal learning signals but failed to achieve meaningful control in practice. This contrast reflects a fundamental trade-off which is that model-free methods tend to be more resilient in sparse-reward and short-horizon tasks, while model-based approaches offer greater data efficiency but require careful tuning of dynamics modeling and planning.

To address these challenges, future work could investigate hybrid approaches that combine Dreamer's imagination-based learning with the robustness of model-free policies. Modifying reward relabeling techniques to better suit short-horizon tasks may also give benefits. Other promising directions include curriculum-based goal selection, exploration enhancements such as intrinsic motivation or prioritized HER sampling to focus on valuable experiences. Additionally, making the world model explicitly goal-aware, or integrating uncertainty-aware planning strategies, can also enhance Dreamer's robustness and generalization in robotic manipulation tasks.

5. Contributions

This work presents a comparative study between a model-free reinforcement learning approach (SAC + HER) and a model-based approach (DreamerV3) on the goal-conditioned FetchReach-v3 robotic task. My main contributions are as follows:

- I checked upon different model-free and model based algorithms which have recently shown great promise.
- I have written the proposal to showcase what experiments we would be doing for this project.
- I have built the base Dreamer algorithm model which uses the open-source implementation of DreamerV3 algorithm and tweaked/adjusted it to work with low-dimensional, goal-conditioned inputs in FetchReach-v3 including a custom HER replay buffer.
- I have performed the implementation of the SAC algorithm as well and fine tuned it to get the final results. Have logged all the results that we could get as benchmark for both these algorithms.

References

[1] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[3] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[4] Marc Peter Deisenroth and Carl Edward Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, 2011.

[5] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning Latent Dynamics for Planning from Pixels. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

[6] Danijar Hafner, Jost Tobias Springenberg, Soroush Nasiriany, Mohammad Babaeizadeh, et al. Mastering Diverse Domains through World Models. In *arXiv preprint arXiv:2301.04104*, 2023.

[7] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Ken Goldberg, and Pieter Abbeel. DayDreamer: World Models for Physical Robot Learning. In *Proceedings of the 6th Conference on Robot Learning (CoRL)*, pages 2226–2240, 2022.

[8] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. DeepMind Control Suite. *arXiv preprint arXiv:1801.00690*, 2018.

[9] Danijar Hafner. DreamerV3: Mastering Long-Horizon Tasks via World Models. *GitHub repository*, `https://github.com/danijar/dreamerv3`.

[10] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Noah Dormann, and René Traore. Stable-Baselines3: Reliable implementations of reinforcement learning algorithms. *GitHub repository*, `https://github.com/DLR-RM/stable-baselines3`.