DayDreamer: World Models for Physical Robot Learning

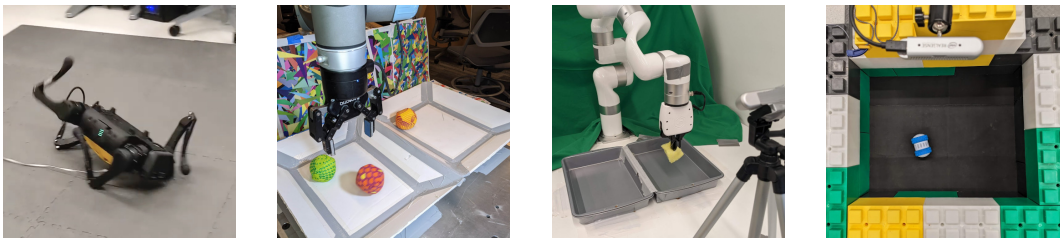
Philipp Wu* **Alejandro Escontrela*** **Danijar Hafner***

Ken Goldberg **Pieter Abbeel**

University of California, Berkeley

*Equal contribution

Abstract: To solve tasks in complex environments, robots need to learn from experience. Deep reinforcement learning is a common approach to robot learning but requires a large amount of trial and error to learn, limiting its deployment in the physical world. As a consequence, many advances in robot learning rely on simulators. On the other hand, learning inside of simulators fails to capture the complexity of the real world, is prone to simulator inaccuracies, and the resulting behaviors do not adapt to changes in the world. The Dreamer algorithm has recently shown great promise for learning from small amounts of interaction by planning within a learned world model, outperforming pure reinforcement learning in video games. Learning a world model to predict the outcomes of potential actions enables planning in imagination, reducing the amount of trial and error needed in the real environment. However, it is unknown whether Dreamer can facilitate faster learning on physical robots. In this paper, we apply Dreamer to 4 robots to learn online and directly in the real world, without any simulators. Dreamer trains a quadruped robot to roll off its back, stand up, and walk from scratch and without resets in only 1 hour. We then push the robot and find that Dreamer adapts within 10 minutes to withstand perturbations or quickly roll over and stand back up. On two different robotic arms, Dreamer learns to pick and place objects from camera images and sparse rewards, approaching human-level teleoperation performance. On a wheeled robot, Dreamer learns to navigate to a goal position purely from camera images, automatically resolving ambiguity about the robot orientation. Using the same hyperparameters across all experiments, we find that Dreamer is capable of online learning in the real world, which establishes a strong baseline. We release our infrastructure for future applications of world models to robot learning. Videos are available on the project website: https://danijar.com/daydreamer



(a) A1 Quadruped Walking (b) UR5 Visual Pick Place (c) XArm Visual Pick Place (d) Sphero Navigation

Figure 1: To study the applicability of Dreamer for sample-efficient robot learning, we apply the algorithm to learn robot locomotion, manipulation, and navigation tasks from scratch in the real world on 4 robots, without simulators. The tasks evaluate a diverse range of challenges, including continuous and discrete actions, dense and sparse rewards, proprioceptive and camera inputs, as well as sensor fusion of multiple input modalities. Learning successfully using the same hyperparameters across all experiments, Dreamer establishes a strong baseline for real world robot learning.

1 Introduction

Teaching robots to solve complex tasks in the real world is a foundational problem of robotics research. Deep reinforcement learning (RL) offers a popular approach to robot learning that enables robots to improve their behavior over time through trial and error. However, current algorithms require too much interaction with the environment to learn successful behaviors, making them impractical for many real world tasks. Recently, modern *world models* have shown great promise for data efficient learning in simulated domains and video games (Hafner et al., 2019; 2020). Learning world models from past experience enables robots to imagine the future outcomes of potential actions, reducing the amount of trial and error in the real environment needed to learn successful behaviors.

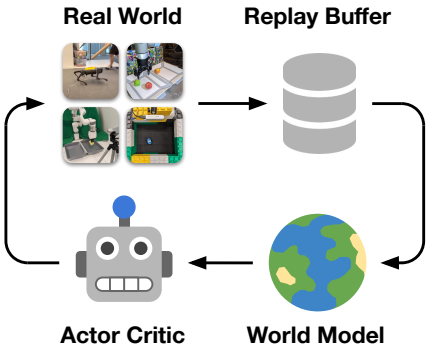
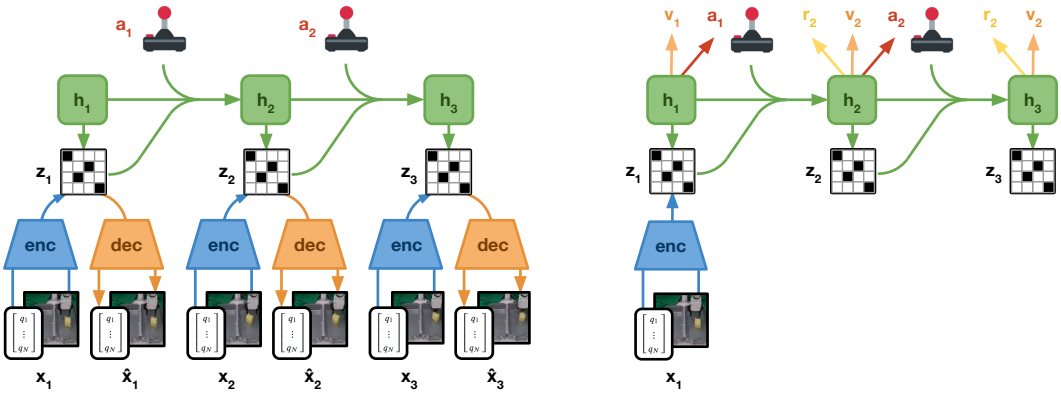


Figure 2: Dreamer follows a simple pipeline for online learning on robot hardware without simulators. The current learned policy collects experience on the robot. This experience is added to the replay buffer. The world model is trained on replayed off-policy sequences through supervised learning. An actor critic algorithm optimizes a neural network policy from imagined rollouts in the latent space of the world model. We parallelize data collection and neural network learning so learning steps can continue while the robot is moving and to enable low-latency action computation.

While learning accurate world models can be challenging, they offer compelling properties for robot learning. By predicting future outcomes, world models allow for planning and behavior learning given only small amounts of real world interaction (Gal et al., 2016; Ebert et al., 2018). Moreover, world models summarize general dynamics knowledge about the environment that, once learned, could be reused for a wide range of downstream tasks (Sekar et al., 2020). World models also learn representations that fuse multiple sensor modalities and integrate them into latent states, reducing the need for sophisticated state estimators. Finally, world models generalize well from available offline data (Yu et al., 2021), which could further accelerate learning in the real world.

Despite the promises of world models, learning accurate world models for the real world is a big open challenge. In this paper, we leverage recent advances of the Dreamer world model for training a variety of robots in the most straight-forward and fundamental problem setting: online reinforcement learning in the real world, without simulators or demonstrations. As shown in Figure 2, Dreamer learns a world model from a replay buffer of past experience, learns behaviors from rollouts imagined in the latent space of the world model, and continuously interacts with the environment to explore and improve its behaviors. Our aim is to push the limits of robot learning directly in the real world and offer a robust platform to enable future work that develops the benefits of world models for robot learning. The key contributions of this paper are summarized as follows:

- **Dreamer on Robots** We apply Dreamer to 4 robots, demonstrating successful learning directly in the real world, without introducing new algorithms. The tasks cover a range of challenges, including different action spaces, sensory modalities, and reward structures.
- **Walking in 1 Hour** We teach a quadruped from scratch in the real world to roll off its back, stand up, and walk in only 1 hour. Afterwards, we find that the robot adapts to being pushed within 10 minutes, learning to withstand pushes or quickly roll over and get back on its feet.
- **Visual Pick and Place** We train robotic arms to pick and place objects from sparse rewards, which requires localizing objects from pixels and fusing images with proprioceptive inputs. The learned behavior outperforms model-free agents and approaches the performance of a human teleoperator using the same control interface as the robot.
- **Open Source** We publicly release the software infrastructure for all our experiments, which supports different action spaces and sensory modalities, offering a flexible platform for future research of world models for robot learning in the real world.



(a) World Model Learning (b) Behavior Learning

Figure 3: **Neural Network Training** We leverage the Dreamer algorithm (Hafner et al., 2019; 2020) for fast robot learning in real world. Dreamer consists of two main neural network components, the world model and the policy. **Left:** The world model follows the structure of a deep Kalman filter that is trained on subsequences drawn from the replay buffer. The **encoder** fuses all sensory modalities into discrete codes. The decoder reconstructs the inputs from the codes, providing a rich learning signal and enabling human inspection of model predictions. A **recurrent state-space model** (RSSM) is trained to predict future codes given actions, without observing intermediate inputs. **Right:** The world model enables massively parallel policy optimization from imagined rollouts in the compact latent space using a large batch size, without having to reconstruct sensory inputs. Dreamer trains a **policy network** and **value network** from the imagined rollouts and a learned **reward function**.

2 Approach

We leverage the Dreamer algorithm (Hafner et al., 2019; 2020) for online learning on physical robots, without the need for simulators. This section summarizes the general algorithm, as well as details on the training architecture and sensor fusion needed for the robotics experiments. Figure 2 shows an overview of the approach. Dreamer learns a world model from a replay buffer of past experiences, uses an actor critic algorithm to learn behaviors from trajectories predicted by the learned model, and deploys its behavior in the environment to continuously grow the replay buffer. We decouple learning updates from data collection to meet latency requirements and to enable fast training without waiting for the environment. In our implementation, a learner thread continuously trains the world model and actor critic behavior, while an actor thread in parallel computes actions for environment interaction.

World Model Learning The world model is a deep neural network that learns to predict the environment dynamics, as shown in Figure 3 (left). Because sensory inputs can be large images, we predict future representations rather than future inputs. This reduces accumulating errors and enables massively parallel training with a large batch size. Thus, the world model can be thought of as a fast simulator of the environment that the robot learns autonomously, starting from a blank slate and continuously improving its model as it explores the real world. The world model is based on the Recurrent State-Space Model (RSSM; Hafner et al., 2018), which consists of four components:

$$\begin{aligned} &\text{Encoder Network:} && \mathrm{enc}_\theta(s_t \mid s_{t-1}, a_{t-1}, x_t) && \text{Decoder Network:} && \mathrm{dec}_\theta(s_t) \approx x_t \\ &\text{Dynamics Network:} && \mathrm{dyn}_\theta(s_t \mid s_{t-1}, a_{t-1}) && \text{Reward Network:} && \mathrm{rew}_\theta(s_{t+1}) \approx r_t \end{aligned} \tag{1}$$

Physical robots are often equipped with multiple sensors of different modalities, such as proprioceptive joint readings, force sensors, and high-dimensional inputs such as RGB and depth camera images. The encoder network fuses all sensory inputs x_t together into the stochastic representations z_t. The dynamics model learns to predict the sequence of stochastic representations by using its recurrent state h_t. The decoder reconstructs the sensory inputs to provide a rich signal for learning representations and enables human inspection of model predictions, but is not needed while learning behaviors from latent rollouts. In our experiments, the robot has to discover task rewards by interacting with the real world, which the reward network learns to predict. Using manually specified rewards as a function of the decoded sensory inputs is also possible. We optimize all components of the world model jointly by stochastic backpropagation (Kingma and Welling, 2013; Rezende et al., 2014).

Actor Critic Learning While the world model represents task-agnostic knowledge about the dynamics, the actor critic algorithm learns a behavior that is specific to the task at hand. As shown in Figure 3 (right), we learn behaviors from rollouts that are predicted in the latent space of the world model, without decoding observations. This enables massively parallel behavior learning with typical batch sizes of 16K on a single GPU, similar to specialized modern simulators (Makoviychuk et al., 2021). The actor critic algorithm consists of two neural networks:

$$\text{Actor Network:} \quad \pi(a_t \mid s_t) \qquad \text{Critic Network:} \quad v(s_t) \tag{2}$$

The role of the actor network is to learn a distribution over successful actions a_t for each latent model state s_t that maximizes the sum of future predicted task rewards. The critic network learns to predict the sum of future task rewards through temporal difference learning (Sutton and Barto, 2018). This is important because it allows the algorithm to take into account rewards beyond the planning horizon of $H = 16$ steps to learn long-term strategies. Given a predicted trajectory of model states, the critic is trained to regress the return of the trajectory. A simple choice would be to compute the return as the sum of N intermediate rewards plus the critic's own prediction at the next state. To avoid the choice of an arbitrary value for N, we instead compute λ-returns, which average over all $N \in [1, H-1]$ and are computed as follows:

$$V_t^\lambda \doteq r_t + \gamma\Big((1-\lambda)v(s_{t+1}) + \lambda V_{t+1}^\lambda\Big), \quad V_H^\lambda \doteq v(s_H). \tag{3}$$

While the critic network is trained to regress the λ-returns, the actor network is trained to maximize them. Different gradient estimators are available for computing the policy gradient for optimizing the actor, such as Reinforce (Williams, 1992) and the reparameterization trick (Kingma and Welling, 2013; Rezende et al., 2014) that directly backpropagates return gradients through the differentiable dynamics network (Henaff et al., 2019). Following Hafner et al. (2020), we choose reparameterization gradients for continuous control tasks and Reinforce gradients for tasks with discrete actions. In addition to maximizing returns, the actor is also incentivized to maintain high entropy to prevent collapse to a deterministic policy and maintain some amount of exploration throughout training:

$$\mathcal{L}(\pi) \doteq -\operatorname{E}\big[\textstyle\sum_{t=1}^{H} \ln \pi(a_t \mid s_t)\, \operatorname{sg}(V_t^\lambda - v(s_t)) + \eta \operatorname{H}\big[\pi(a_t \mid s_t)\big]\big] \tag{4}$$

We optimize the actor and critic using the Adam optimizer (Kingma and Ba, 2014). To compute the λ-returns, we use a slowly updated copy of the critic network as common in the literature (Mnih et al., 2015; Lillicrap et al., 2015). The actor and critic gradients do not affect the world model, as this would lead to incorrect and overly optimistic model predictions. The hyperparameters are listed in Appendix D. Compared to Hafner et al. (2020), there is no training frequency parameter because the neural networks is optimized in parallel with data collection, without rate limiting.

3 Experiments

We evaluate Dreamer on 4 robots, each with a different task, and compare its performance to appropriate algorithmic and human baselines. The experiments are representative of common robotic tasks, such as locomotion, manipulation, and navigation. The tasks pose a diverse range of challenges, including continuous and discrete actions, dense and sparse rewards, proprioceptive and image observations, and sensor fusion. Learned world models have various properties that make them well suited for robot learning. The goal of the experiments is to evaluate whether the recent successes of learned world models enables sample-efficient robot learning directly in the real world. Specifically, we aim to answer the following research questions:

- Does Dreamer enable robot learning directly in the real world, without simulators?
- Does Dreamer succeed across various robot platforms, sensory modalities, and action spaces?
- How does the data-efficiency of Dreamer compare to previous reinforcement learning algorithms?

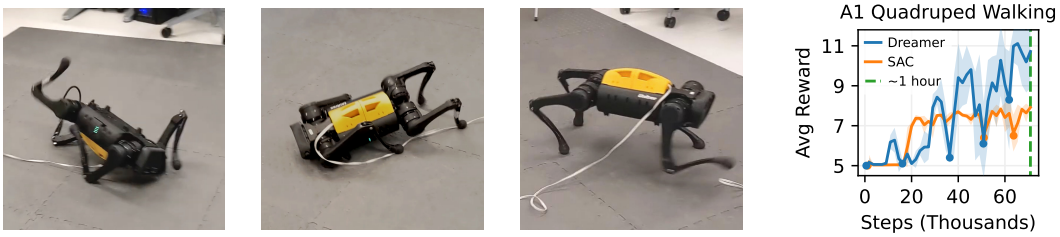


Figure 4: **A1 Quadruped Walking** Starting from lying on its back with the feet in the air, Dreamer learns to roll over, stand up, and walk in 1 hour of real world training time, without simulators or resets. In contrast, SAC only learns to roll over but neither to stand up nor to walk. For SAC, we also had to help the robot out of a dead-locked leg configuration during training. On the right we show training curves for both SAC and Dreamer. The maximum reward is 14. The filled circles indicate times where the robot fell on its back, requiring the learning of a robust strategy for getting back up. After 1 hour of training, we start pushing the robot and find that it adapts its behavior within 10 minutes to withstand light pushes and quickly roll back on its feet for hard pushes. The graph shows a single training run with the shaded area indicating one standard deviation within each time bin.

Implementation We build on the official implementation of DreamerV2 (Hafner et al., 2020), which handles multiple sensory modalities. We develop an asynchronous actor and learner setup, which is essential in environments with high control rates, such as the quadruped, and also accelerates learning for slower environments, such as the robot arms. The actor thread computes online actions for the robot and sends trajectories of 128 time steps to the replay buffer using ZeroMQ. The learner thread samples data from the replay buffer, updates the world model, and optimizes the policy using imagination rollouts. Policy weights are synced from the learner to the actor every 20 seconds. We use an RSSM with 256 units to speed up the training computation. We use identical hyperparameters across all experiments, enabling off-the-shelf training on different robot embodiments.

Baselines We compare to a strong learning algorithm for each of our experimental setups. The A1 quadruped robot uses continuous actions and low-dimensional inputs, allowing us to compare to SAC (Haarnoja et al., 2018a;b), a popular algorithm for data-efficient continuous control. For the visual pick and place experiments on the XArm and UR5 robots, inputs are images and proprioceptive readings and actions are discrete, suggesting algorithms from the DQN (Mnih et al., 2015) line of work as baselines. We choose Rainbow (Hessel et al., 2018) as a powerful representative of this category, an algorithm that combines many improvements of DQN. To input the proprioceptive readings, we concatenate them as broadcasted planes to the RGB channels of the image, a common practice in the literature (Schrittwieser et al., 2019). For the UR5, we additionally compare against PPO (Schulman et al., 2017), with similar modifications for fusing image and proprioceptive readings. In addition, we compare against a human operator controlling the robot arm through the robot control interface, which provides an strong baseline for the robot performance. For the Sphero navigation task, inputs are images and actions are continuous. The state-of-the-art baseline in this category is DrQv2 (Yarats et al., 2021), which uses image augmentation to increase sample-efficiency.

3.1 A1 Quadruped Walking

This high-dimensional continuous control task requires training a quadruped robot to roll over from its back, stand up, and walk forward at a fixed target velocity. Prior work in quadruped locomotion requires either extensive training in simulation under domain randomization, using recovery controllers to avoid unsafe states, or defining the action space as parameterized trajectory generators that restrict the space of motions (Rusu et al., 2016; Peng et al., 2018; Rudin et al., 2021; Lee et al., 2020; Yang et al., 2019). In contrast, we train in the end-to-end reinforcement learning setting directly on the robot, without simulators or resets. We use the Unitree A1 robot that consists of 12 direct drive motors. The motors are controlled at 20 Hz via continuous

Figure 8: Within 10 minutes of perturbing the learned walking behavior, the robot adapts to withstanding pushes or quickly rolling over and back on its feet.

Figure 5: **UR5 Multi Object Visual Pick and Place** This task requires learning to locate three ball objects from third-person camera images, grasp them, and move them into the other bin. The arm is free to move within and above the bins and sparse rewards are given for grasping a ball and for dropping it in the opposite bin. The environment requires the world model to learn multi-object dynamics in the real world and the sparse reward structure poses a challenge for policy optimization. Dreamer overcomes the challenges of visual localization and sparse rewards on this task, learning a successful strategy within a few hours of autonomous operation.

actions that represent motor angles that are realized by a PD controller on the hardware. The input consists of motor angles, orientations, and angular velocities. To protect the motors from high-frequency action oscillations generated by the RL policy, we filter out high-frequency motor commands through a Butterworth filter. Due to space constraints, we manually intervene when the robot has reached the end of the available training area, without modifying the joint configuration or orientation that the robot is in.

The reward function is the sum of five terms. An upright reward is computed from the base frame up vector $\hat{z}^T$, terms for matching the standing pose are computed from the joint angles of the hips, shoulders, and knees, and a forward velocity term is computed from the projected forward velocity ${}^{\mathcal{B}}v^x$ and the total velocity ${}^{\mathcal{B}}v$. Without the reward curriculum, the agent receives spurious reward values due to the velocity estimator's dependence on foot-ground contact events. Each of the five terms is active while its preceding terms are satisfied to at least 0.7 and otherwise set to 0:

$$\begin{aligned} r^{\text{upr}} &\doteq (\hat{z}^T[0,0,1]\text{-}1)/2 \quad r^{\text{hip}} \doteq 1\text{-}\tfrac{1}{4}\|\, q^{\text{hip}} + 0.2\,\|_1 \quad r^{\text{shoulder}} \doteq 1\text{-}\tfrac{1}{4}\|\, q^{\text{shoulder}} + 0.2\,\|_1 \\ r^{\text{knee}} &\doteq 1\text{-}\tfrac{1}{4}\|\, q^{\text{knee}}\text{-}1.0\,\|_1 \quad r^{\text{velocity}} \doteq 5\big(\max(0, {}^{\mathcal{B}}v_x)/\|{}^{\mathcal{B}}v\|_2 \cdot \text{clip}({}^{\mathcal{B}}v_x/0.3, \text{-}1, 1) + 1\big) \end{aligned} \tag{5}$$

As shown in Figure 4, after one hour of training, Dreamer learns to consistently flip the robot over from its back, stand up, and walk forward. In the first 5 minutes of training, the robot manages to roll off its back and land on its feet. 20 minutes later, it learns how to stand up on its feet. About 1 hour into training, the robot learns a pronking gait to walk forward at the desired velocity. After succeeding at this task, we tested the robustness of the algorithms by repeatedly knocking the robot off of its feet with a large pole, shown in Figure 8. Within 10 minutes of additional online learning, the robot adapts and withstand pushes or quickly rolls back on its feet. In comparison, SAC quickly learns to roll off its back but fails to stand up or walk given the small data budget.

3.2 UR5 Multi-Object Visual Pick and Place

Common in warehouse and logistics environments, pick and place tasks require a robot manipulator to transport items from one bin into another. Figure 5 shows a successful pick and place cycle of this task. The task is challenging because of sparse rewards, the need to infer object positions from pixels, and the challenging dynamics of multiple moving objects. The sensory inputs consist of proprioceptive readings (joint angles, gripper position, end effector Cartesian position) and a 3rd person RGB image of the scene. Successfully grasping one of the 3 objects, detected by partial gripper closure, results in a $+1$ reward, releasing the object in the same bin gives a -1 reward, and placing in the opposite bin gives a $+10$ reward. We control the UR5 robot from Universal Robotics at 2 Hz. Actions are discrete for moving the end effector in increments along X, Y, and Z axes and for toggling the gripper state. Movement in the Z axis is only enabled while holding an object and the gripper automatically opens once above the correct bin. We estimate human teleoperation performance by recording 3 demonstrators for 20 minutes each, controlling the UR5 with a joystick.

Dreamer reaches an average pick rate of 2.5 objects per minute within 8 hours. The robot initially struggles to learn as the reward signal is very sparse, but begins to gradually improve after 2 hours of

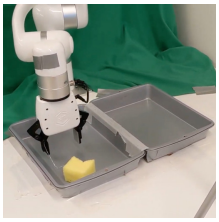
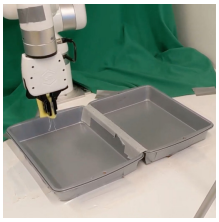
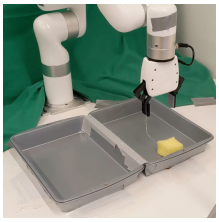
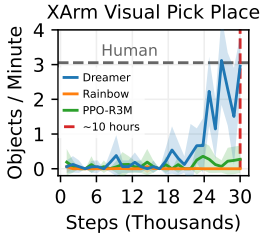





Figure 6: **XArm Visual Pick and Place** The XArm is an affordable robot arm that operates slower than the UR5. To demonstrate successful learning on this robot, we use a third-person RealSense camera with RGB and depth modalities, as well as proprioceptive inputs for the robot arm, requiring the world model to learn sensor fusion. The pick and place task uses a soft object. While soft objects would be challenging to model accurately in a simulator, Dreamer avoids this issue by directly learning on the real robot without a simulator. While Rainbow converges to the local optimum of grasping and ungrasping the object in the same bin, Dreamer learns a successful pick and place policy from sparse rewards in under 10 hours.

training. The robot first learns to localize the objects and toggles the gripper when near an object. Over time, grasping becomes precise and the robot learns to push objects out of corners. Figure 5 shows the learning curves of Dreamer compared to Rainbow DQN, PPO, and the human baseline. Both Rainbow DQN and PPO only learn the short-sighted behavior of grasping and immediately dropping objects in the same bin. In contrast, Dreamer approaches human-level teleoperation performance after 8 hours. We hypothesize that Rainbow DQN and PPO fail because they require larger amounts of experience, which is not feasible for us to collect in the real world.

3.3 XArm Visual Pick and Place

While the UR5 robot is a high performance industrial robot, the XArm is an accessible low-cost 7 DOF manipulation, which we control at approximately 0.5 Hz. Similar to Section 3.2, the task requires localizing and grasping a soft object and moving it from one bin to another and back, shown in Figure 6. Because the bins are not slanted, we connect the object to the gripper with a string. This makes it less likely for the object to get stuck in corners at the cost of more complex dynamics. The sparse reward, discrete action space, and observation space match the UR5 setup except for the addition of depth image observations.

Dreamer learns a policy that enables the XArm to achieve an average pick rate of 3.1 objects per minute in 10 hours of time, which is comparable to human performance on this task. Figure 6 shows that Dreamer learns to solve the task within 10 hours, whereas the Rainbow algorithm, a top model-free algorithm for discrete control from pixels, fails to learn. Interestingly, we observed that Dreamer learns to sometimes use the string to pull the object out of a corner before grasping it, demonstrating multi-modal behaviors. Moreover, we observed that when lighting conditions change drastically (such as sharp shadows during sunrise), performance initially collapses but Dreamer then adapts to the changing conditions and exceeds its previous performance after a few hours of additional training, reported in Appendix A.

3.4 Sphero Navigation

We evaluate Dreamer on a visual navigation task that requires maneuvering a wheeled robot to a fixed goal location given only RGB images as input. We use the Sphero Ollie robot, a cylindrical robot with two controllable motors, which we control through continuous torque commands at 2 Hz. Because the robot is symmetric and the robot only has access to image observations, it has to infer the heading direction from the history of observations. The robot is provided with a dense reward equal to the negative L2 distance, which is computed using a oracle vision pipeline that detects the Sphero's position (this information is not provided to the agent). As the goal is fixed, after 100 environment steps, we end the episode and randomize the robot's position through a sequence of high power random motor actions.

In 2 hours, Dreamer learns to quickly and consistently navigate to the goal and stay near the goal for the remainder of the episode. As shown in Figure 7, Dreamer achieves an average distance to the goal of 0.15, measured in units of the area size and averaged across time steps. We find that DrQv2,

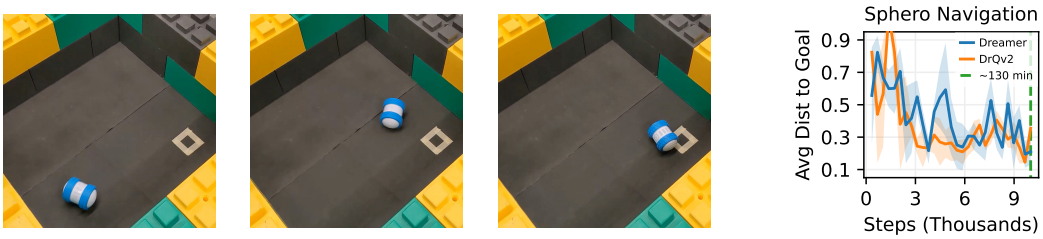


Figure 7: **Sphero Navigation** This task requires the Sphero Ollie robot to navigate to a fixed goal location through continuous actions given a top-down RGB image as the only sensory input. The task requires the robot to localize itself from pixels without proprioceptive inputs, to infer its orientation from the sequence of past images because it is ambiguous from a single image, and to control the robot from under-actuated motors that require building up momentum over time. Dreamer learns a successful policy on this task in under 2 hours.

a model-free algorithm specifically designed to continuous control from pixels, achieves similar performance. This result matches the simulated experiments of Yarats et al. (2021) that showed the two algorithms to perform similarly for continuous control tasks from images.

4 Related Work

Existing work on robot learning commonly leverages large amounts of simulated experience under domain and dynamics randomization before deploying to the real world (Rusu et al., 2016; Peng et al., 2018; OpenAI et al., 2018; Lee et al., 2020; Irpan et al., 2020; Rudin et al., 2021; Kumar et al., 2021; Siekmann et al., 2021; Smith et al., 2021; Escontrela et al., 2022; Miki et al., 2022), leverage fleets of robots to collect experience datasets (Kalashnikov et al., 2018; Levine et al., 2018; Dasari et al., 2019; Kalashnikov et al., 2021; Ebert et al., 2021), or rely on external information such as human expert demonstrations or task priors to achieve sample-efficient learning (Xie et al., 2019; Schoettler et al., 2019; James et al., 2021; Shah and Levine, 2022; Bohez et al., 2022; Sivakumar et al., 2022). However, designing simulated tasks and collecting expert demonstrations is time-consuming. Moreover, many of these approaches require specialized algorithms for leveraging offline experience, demonstrations, or simulator inaccuracies. In contrast, our experiments show that learning end-to-end from rewards in the physical world is feasible for a diverse range of tasks through world models.

Relatively few works have demonstrated end-to-end learning from scratch in the physical world. Visual Foresight (Finn et al., 2016; Finn and Levine, 2017; Ebert et al., 2018) learns a video prediction model to solve real world tasks by online planning, but is limited to short-horizon tasks and requires generating images during planning, making it computationally expensive. In comparison, we learn latent dynamics that enable efficient policy optimization with a large batch size in the compact latent space. Yang et al. (2019; 2022) learn quadruped locomotion through a model-based approach by predicting foot placement and leveraging a domain-specific controller to achieve them. Ha et al. (2020) learn a quadruped walking policy by relying on a scripted reset policy, so the robot does not have to learn to stand up. SOLAR (Zhang et al., 2019) learns a latent dynamics model from images and demonstrates reaching and pushing with a robot arm. Nagabandi et al. (2019) learns manipulation policies by planning through a learned dynamics model from state observations. In comparison, our experiments show successful learning across 4 challenging robot tasks that cover a wide range of challenges and sensory modalities, with a single learning algorithm and hyperparameter setting.

5 Discussion

We applied Dreamer to physical robot learning, finding that modern world models enable sample-efficient robot learning for a range of tasks, from scratch in the real world and without simulators. We also find that the approach is generally applicable in that it can solve robot locomotion, manipulation, and navigation tasks without changing hyperparameters. Dreamer taught a quadruped robot to roll off the back, stand up, and walk in 1 hour from scratch, which previously required extensive training in simulation followed by transfer to the real world or parameterized trajectory generators and given reset policies. We also demonstrate learning to pick and place objects from pixels and sparse rewards on two robot arms in 8–10 hours.

Limitations While Dreamer shows promising results, learning on hardware over many hours creates wear on robots that may require human intervention or repair. Additionally, more work is required to explore the limits of Dreamer and our baselines by training for a longer time. Finally, we see tackling more challenging tasks, potentially by combining the benefits of fast real world learning with those of simulators, as an impactful future research direction.

Acknowledgements We thank Stephen James and Justin Kerr for helpful suggestions and help with printing the protective shell of the quadruped robot. We thank Ademi Adeniji for help with setting up the XArm robot and Raven Huang for help with setting up the UR5 robot. This work was supported in part by an NSF Fellowship, NSF NRI #2024675, and the Vanier Canada Graduate Scholarship.

References

D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

Y. Gal, R. McAllister, and C. E. Rasmussen. Improving pilco with bayesian neural network dynamics models. In *Data-Efficient Machine Learning workshop, ICML*, 2016.

F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, pages 8583–8592. PMLR, 2020.

T. Yu, A. Kumar, R. Rafailov, A. Rajeswaran, S. Levine, and C. Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34: 28954–28967, 2021.

D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

M. Henaff, A. Canziani, and Y. LeCun. Model-predictive policy learning with uncertainty regularization for driving in dense traffic. *arXiv preprint arXiv:1901.02705*, 2019.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018a.

T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265*, 2019.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell. Sim-to-real robot learning from pixels with progressive nets, 2016.

X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, May 2018. doi:10.1109/ICRA.2018.8460528.

N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning, 2021.

J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47), oct 2020. doi:10.1126/scirobotics.abc5986. URL https://doi.org/10.1126%2Fscirobotics.abc5986.

Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani. Data efficient reinforcement learning for legged robots, 2019.

OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation, 2018.

A. Irpan, C. Harris, J. Ibarz, K. Rao, M. Khansari, and S. Levine. Rl-cyclegan: Improving deep-rl robotics with simulation-to-real. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2020)*, 2020.

A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots, 2021.

J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst. Blind bipedal stair traversal via sim-to-real reinforcement learning, 2021.

L. Smith, J. C. Kew, X. B. Peng, S. Ha, J. Tan, and S. Levine. Legged robots that keep on learning: Fine-tuning locomotion policies in the real world, 2021.

A. Escontrela, X. B. Peng, W. Yu, T. Zhang, A. Iscen, K. Goldberg, and P. Abbeel. Adversarial motion priors make good substitutes for complex reward functions, 2022.

T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62), jan 2022. doi:10.1126/scirobotics.abk2822.

D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation, 2018.

S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.

S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn. Robonet: Large-scale multi-robot learning, 2019.

D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale, 2021.

F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets, 2021.

A. Xie, F. Ebert, S. Levine, and C. Finn. Improvisation through physical understanding: Using novel objects as tools with visual foresight. *arXiv preprint arXiv:1904.05538*, 2019.

G. Schoettler, A. Nair, J. Luo, S. Bahl, J. A. Ojea, E. Solowjow, and S. Levine. Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards, 2019.

S. James, K. Wada, T. Laidlow, and A. J. Davison. Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation, 2021.

D. Shah and S. Levine. Viking: Vision-based kilometer-scale navigation with geographic hints, 2022.

S. Bohez, S. Tunyasuvunakool, P. Brakel, F. Sadeghi, L. Hasenclever, Y. Tassa, E. Parisotto, J. Humplik, T. Haarnoja, R. Hafner, M. Wulfmeier, M. Neunert, B. Moran, N. Siegel, A. Huber, F. Romano, N. Batchelor, F. Casarini, J. Merel, R. Hadsell, and N. Heess. Imitate and repurpose: Learning reusable robot movement skills from human and animal behaviors, 2022.

A. Sivakumar, K. Shaw, and D. Pathak. Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube, 2022.

C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in neural information processing systems*, pages 64–72, 2016.

C. Finn and S. Levine. Deep visual foresight for planning robot motion. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2786–2793. IEEE, 2017.

Y. Yang, T. Zhang, E. Coumans, J. Tan, and B. Boots. Fast and efficient locomotion via learned gait transitions. In *Conference on Robot Learning*, pages 773–783. PMLR, 2022.

S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan. Learning to walk in the real world with minimal human effort. *arXiv preprint arXiv:2002.08550*, 2020.

M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. Johnson, and S. Levine. Solar: deep structured representations for model-based reinforcement learning. In *International Conference on Machine Learning*, 2019.

A. Nagabandi, K. Konoglie, S. Levine, and V. Kumar. Deep dynamics models for learning dexterous manipulation, 2019.

G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. ISSN 0893-6080.

T.-Y. Yang, T. Zhang, L. Luu, S. Ha, J. Tan, and W. Yu. Safe reinforcement learning for legged locomotion, 2022. URL https://arxiv.org/abs/2203.02638.

S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan. Learning to walk in the real world with minimal human effort, 2020. URL https://arxiv.org/abs/2002.08550.

L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours, 2015.

H. Ha and S. Song. Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding. *Conference on Robot Learning*, 2021.

S. James and A. J. Davison. Q-attention: Enabling efficient learning for vision-based robotic manipulation, 2021.

E. Tzeng, C. Devin, J. Hoffman, C. Finn, P. Abbeel, S. Levine, K. Saenko, and T. Darrell. Adapting deep visuomotor representations with weak pairwise constraints, 2015.

I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

M. P. Deisenroth, G. Neumann, J. Peters, et al. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1–2):1–142, 2013.

K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pages 4754–4765, 2018.

A. Nagabandi, G. Yang, T. Asmar, R. Pandya, G. Kahn, S. Levine, and R. S. Fearing. Learning image-conditioned dynamics models for control of under-actuated legged millirobots, 2017.

P. Becker-Ehmck, M. Karl, J. Peters, and P. van der Smagt. Learning to fly via deep model-based reinforcement learning. *arXiv preprint arXiv:2003.08876*, 2020.

F. Deng, I. Jang, and S. Ahn. Dreamerpro: Reconstruction-free model-based reinforcement learning with prototypical representations. *arXiv preprint arXiv:2110.14565*, 2021.

M. Okada and T. Taniguchi. Dreaming: Model-based reinforcement learning by latent imagination without reconstruction. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4209–4215. IEEE, 2021.

H. Bharadhwaj, M. Babaeizadeh, D. Erhan, and S. Levine. Information prioritization through empowerment in visual model-based rl. *arXiv preprint arXiv:2204.08585*, 2022.

K. Paster, L. E. McKinney, S. A. McIlraith, and J. Ba. Blast: Latent dynamics models from bootstrapping. In *Deep RL Workshop NeurIPS 2021*, 2021.

K. Hsu, M. J. Kim, R. Rafailov, J. Wu, and C. Finn. Vision-based manipulators need to also see from their hands, 2022. URL https://arxiv.org/abs/2203.12677.

A Adaptation

Real world robot learning faces practical challenges such as changing environmental conditions and time varying dynamics. We found that Dreamer is able to adapt to the current environmental conditions with no change to the learning algorithm. This shows promise for using Dreamer in continual learning settings (Parisi et al., 2019). Adaptation of the quadruped to external perturbations is reported in Section 3.1 and Figure 8.

The XArm, situated near large windows, is able to adapt and maintain performance under the presence of changing lighting conditions. The XArm experiments were conducted after sundown to keep the lighting conditions constant throughout training. Figure A.1 shows the learning curve of the XArm. As expected, the performance of the XArm drops during sunrise. However, the XArm is able to adapt to the change in lighting conditions in about 5 hours time and recover the original performance, which is faster than it would be to train from scratch. A careful inspection of the image observations at these times, as shown in Figure A.1, reveals that the robot received observations with strong light rays covering the scene which greatly differs from the original training observations.

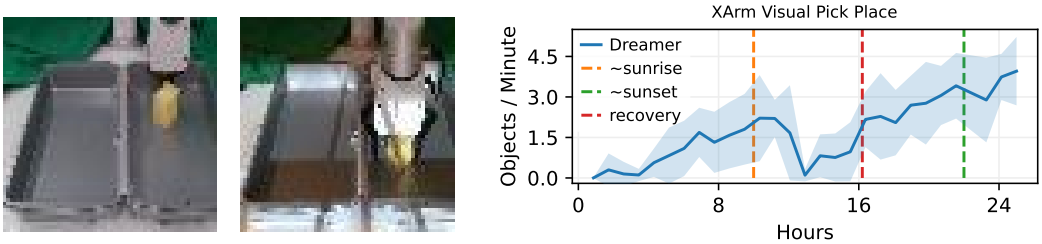


Figure A.1: The left two images are raw observations consumed by Dreamer. The leftmost image is an image observation as seen by the XArm at night, when it was trained. The next image shows an observation during sunrise. Despite the vast difference in pixel space, the XArm is able to recover, and then surpass, the original performance in approximately 5 hours. Even after 24 hours when the lighting shifts to night time conditions, the XArm is able to maintain performance.

B Imagination

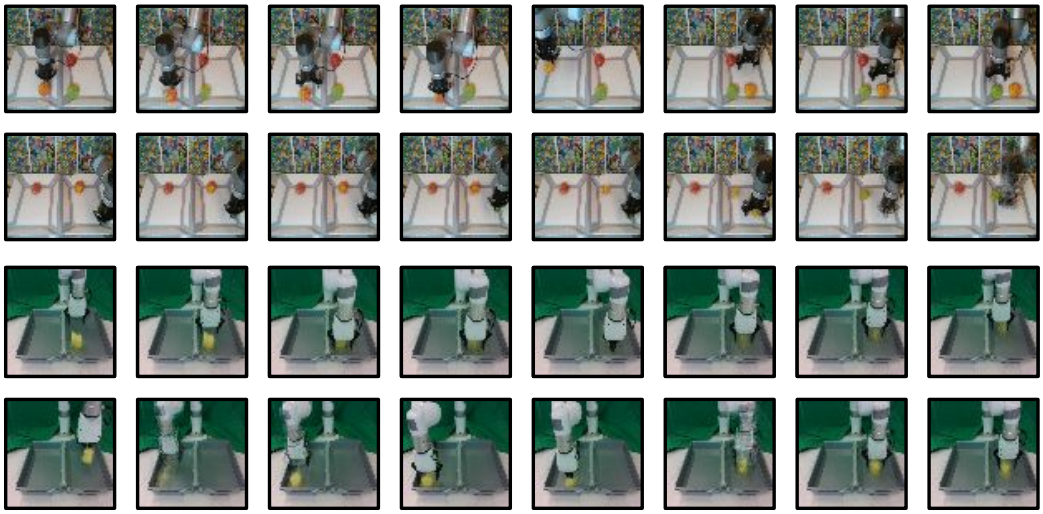


Figure B.1: To introspect the policy, we can roll out trajectories in the latent space of Dreamer, then decode the images to visualize the intent of the actor network. Each row is an imagined trajectory, showing every 2nd frame. **Top:** Latent rollouts on the UR5 environment. Multiple objects introduce more visual complexity that the network has to model. Note the second trajectory, which shows a static orange ball becoming a green ball. **Bottom:** Latent rollouts on the XArm environment.

C Detailed Related Work

RL for locomotion A common approach is to train RL agents from large amounts of simulated data under domain and dynamics randomization (Peng et al., 2018; Lee et al., 2020; Rudin et al., 2021; Siekmann et al., 2021; Escontrela et al., 2022; Miki et al., 2022; Kumar et al., 2021; Rusu et al., 2016; Bohez et al., 2022), then freezing the learned policy and deploying it to the real world. Smith et al. (2021) explored pre-training policies in simulation and fine-tuning them with real world data. Yang et al. (2019) investigate learning a dynamics model using a multi-step loss and using model predictive control to accomplish a specified task. Yang et al. (2022) train locomotion policies in the real world but require a recovery controller trained in simulation to avoid unsafe states. In contrast, we use no simulators or reset policies and directly train on the physical robot. While prior work in locomotion has successfully learned walking behaviors in the real world, these works generally required several domain-specific assumptions or pretraining with simulators. Ha et al. (2020) achieved successful walking on the Minitaur robot in 90 minutes. However, the authors manually programmed a reset policy that was used when the robot fell on its back, while in our work the robot must learn to flip over and stand up. Additionally, the Minitaur robot is simpler than the A1 as it has 8 actuators compared to 12 on the A1.

RL for manipulation Learning promises to enable robot manipulators to solve contact rich tasks in open real world environments. One class of methods attempts to scale up experience collection through a fleet of robots (Kalashnikov et al., 2018; 2021; Ebert et al., 2021; Dasari et al., 2019; Levine et al., 2018). In contrast, we only leverage one robot, but parallelize an agent's experience by using the learned world model. Another common approach is to leverage expert demonstrations or other task priors (Pinto and Gupta, 2015; Ha and Song, 2021; Xie et al., 2019; Schoettler et al., 2019; Sivakumar et al., 2022). James and Davison (2021); James et al. (2021) leverages a few demonstrations to increase the sample-efficiency of Q learning by focusing the learner on important aspects of the scene. Other approaches, as in locomotion, first utilize a simulator, then transfer to the real world (Tzeng et al., 2015; Akkaya et al., 2019; OpenAI et al., 2018; Irpan et al., 2020). Our work focuses on single-robot environments where the agent must learn through a small amount of interaction with the world. Meanwhile, the Google Arm Farm line of work by Levine et al. leverages over 580k grasp attempts gathered by 7 robots and collected over 4 months. We believe that a method such as Dreamer could benefit greatly from this scale of training data, however it is unlikely that works such as MT-OPT/QT-OPT Kalashnikov et al. (2018; 2021) would work well in the low data regime that Dreamer excels in.

Model-based RL Due to its higher sample-efficiency over model-free methods, model-based RL is a promising approach to learning on real world robots (Deisenroth et al., 2013). A model based method first learns a dynamics model, which can then be used to plan actions (Nagabandi et al., 2019; Hafner et al., 2018; Chua et al., 2018; Nagabandi et al., 2017; Becker-Ehmck et al., 2020), or be used as a simulator to learn a policy network as in Dreamer (Hafner et al., 2019; 2020). One approach to tackle the high visual complexity of the world is to learn an action conditioned video prediction model (Finn and Levine, 2017; Ebert et al., 2018; Finn et al., 2016). One downside of this approach is the need to directly predict high dimensional observations, which can be computationally inefficient and easily drift. Dreamer learns a dynamics model in a latent space, allowing more efficient rollouts and avoids relying on high quality visual reconstructions for the policy. Another line of work proposes to learn latent dynamics models without having to reconstruct inputs (Deng et al., 2021; Okada and Taniguchi, 2021; Bharadhwaj et al., 2022; Paster et al., 2021), which we see as a promising approach for supporting moving view points in cluttered environments.

D Hyperparameters

Name	Symbol	Value
General		
Replay capacity (FIFO)	—	10^6
Start learning	—	10^4
Batch size	B	32
Batch length	T	32
MLP size	—	4×512
Activation	—	$\text{LayerNorm} + \text{ELU}$
World Model		
RSSM size	—	512
Number of latents	—	32
Classes per latent	—	32
KL balancing	—	0.8
Actor Critic		
Imagination horizon	H	15
Discount	γ	0.95
Return lambda	λ	0.95
Target update interval	—	100
All Optimizers		
Gradient clipping	—	100
Learning rate	—	10^{-4}
Adam epsilon	ϵ	10^{-6}

E Environment and Hardware Details

For every robot setup that involved vision (UR5, XArm, Sphero), we used a RealSense D435 camera positioned to offer a fixed 3rd person view of the scene.

A1 We used the A1 quadrupedal robot by Unitree. The RL policy outputs actions at a frequency that is too high for the PD controller to track, which we overcome by lowpass filtering the action sequence. The joint range allows the legs to self-collide with the body, which can be damaging to the motors and increase battery consumption. We limited the joint range to decrease self-collisions. Finally, the EKF velocity estimator relies on foot-ground contact events to prevent significant drift in the estimates, so we employ a curriculum reward function that does not reward the robot for forward velocity until the robot is upright with extended legs. We also designed a shell which we 3D printed in order to better protect the cables and hardware and provide a smoother rolling over.

XArm & UR5 We utilized slanted bins to prevent objects from leaving the work area during the long-running pick and place experiments on the UR5, which is common practice Levine et al. (2018); Kalashnikov et al. (2018). We also added a partition behind the setup to keep the background constant. It would be interesting to study how a gripper-mounted camera would impact policy performance Hsu et al. (2022), however we report strong results without this design choice. For the XArm we use the uFactory xArm Gripper. For the UR5, we use the Robotiq 2F-85 parallel jaw gripper. The bin locations are predetermined and provided as part of the environment to prevent the robot from colliding with the bin. In addition, movement in the Z axis is only enabled while holding an object and the gripper automatically opens once above the other bin.

Sphero We used a rectangular enclosure of $0.8 \times 0.8\text{m}^2$ to keep the sphero robot within the camera view. We used a simple OpenCV script to estimate the L2 distance between the Sphero and the goal position to provide a dense reward for policy optimization. This positional information was not provided to the agent, which had to learn from the raw top-down images.