THE INGREDIENTS OF REAL-WORLD ROBOTIC REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

The success of reinforcement learning in the real world has been limited to instrumented laboratory scenarios, often requiring arduous human supervision to enable continuous learning. In this work, we discuss the required elements of a robotic system that can continually and autonomously improve with data collected in the real world, and propose a particular instantiation of such a system. Subsequently, we investigate a number of challenges of learning without instrumentation – including the lack of episodic resets, state estimation, and hand-engineered rewards – and propose simple, scalable solutions to these challenges. We demonstrate the efficacy of our proposed system on dexterous robotic manipulation tasks in simulation and the real world, and also provide an insightful analysis and ablation study of the challenges associated with this learning paradigm.

1 INTRODUCTION

Reinforcement learning (RL) can in principle enable real-world autonomous systems, such as robots, to autonomously acquire a large repertoire of skills. Perhaps more importantly, reinforcement learning can enable such systems to *continuously improve* the proficiency of their skills from experience. However, realizing this promise in reality has proven challenging: even with reinforcement learning methods that can acquire complex behaviors from high-dimensional low-level observations, such as images, the typical assumptions of the reinforcement learning problem setting do not fit perfectly into the constraints of the real world. For this reason, most successful robotic learning experiments have been demonstrated with varying levels of instrumentation, in order to make it practical to define reward functions (e.g. by using auxiliary sensors (Haarnoja et al., 2018a; Kumar et al., 2016; Andrychowicz et al., 2018)), and in order to make it practical to *reset* the environment between trials (e.g. using manually engineered contraptions (Zhu et al., 2019)). In order to really make it practical for autonomous learning systems to improve continuously through real-world operation, we must lift these constraints and design learning systems whose assumptions match the constraints of the real world, and allow for uninterrupted continuous learning with large amounts of real world experience. What exactly is holding back our reinforcement learning algorithms from being deployed for learning robotic tasks (for instance manipulation) directly in the real world?

We hypothesize that our current reinforcement learning algorithms make a number of unrealistic assumptions that make real world deployment challenging – access to low-dimensional Markovian state, known reward functions, and availability of episodic resets. In practice, this means that significant human engineering is required to materialize these assumptions in order to conduct real-world reinforcement learning, which limits the ability of learning-enabled robots to collect large amounts of experience automatically in a variety of naturally occuring environments. Even if we can engineer a complex solution for instrumentation in one environment, the same may need to be done for every environment being learned in. When using deep function approximators, actually collecting large amounts of real world experience is typically crucial for effective generalization. The inability to collect large amounts of real world data autonomously significantly limits the ability of these robots to learn robust, generalizable behaviors. In this work, we propose that overcoming these challenges requires designing robotic systems that possess three fundamental capabilities: (1) they are able to learn from their own raw sensory inputs, (2) they are able to assign rewards to their own behaviors with minimal human intervention, (3) they are able to learn continuously in non-episodic settings without requiring human operators to manually reset the environment. We believe that a system with these capabilities will bring us significantly closer to the goal of continuously improv-

ing robotic agents that leverage large amounts of their own real world experience, without requiring significant human instrumentation and engineering effort.

Having laid out these requirements, we propose a practical instantiation of such a learning system, which afford the above capabilities. While prior works have studied each of these issues in isolation, combining solutions to these issues is non-trivial and results in a particularly challenging learning problem. We provide a detailed empirical analysis of these issues, both in simulation and on a real-world robotic platform, and propose a number of simple but effective solutions that can make it possible to produce a complete robotic learning system that can learn autonomously, handle raw sensory inputs, learn reward functions from easily available supervision, and learn without manually designed reset mechanisms. We show that this system is well suited for learning dexterous robotic manipulation tasks in the real world, and substantially outperforms ablations and prior work. While the individual components that we combine to design our robotic learning system are based heavily on prior work, both the combination of these components and their specific instantiations are novel. Indeed, we show that without the particular design decisions motivated by our experiments, naïve designs that follow prior work generally fail to satisfy one of the three requirements that we lay out.

2 The Structure of a Real-World RL System

Let us start by considering the standard reinforcement learning paradigm, where we operate in a Markov decision process with state space $\mathcal{S}$, action space $\mathcal{A}$, unknown transition dynamics $\mathcal{T}$, unknown reward function $\mathcal{R}$ and an episodic initial state distribution ρ. In RL, the goal is to learn a policy to maximize the expected sum of rewards via environment interactions.

Although this formalism is simple and concise, it does not capture all of the complexities of real-world robotic learning problems. If a robotic system is to learn continuously and autonomously in the real world, we must ensure that it can learn under *all* assumptions that are imposed by the real world. The real world does not have instrumentation available to easily provide low dimensional state estimates, rewards or episodic resets. To move from the idealized MDP formulation to the real world, we require a system that has the following properties. **Firstly**, all of the information necessary for learning must be obtained from the robot's own sensors. This includes all information about the state and necessitates that the policy must be learned from high-dimensional and low-level sensory observations, such as camera images. **Secondly**, the robot must also obtain the *reward signal* itself from its own sensor readings. This is exceptionally difficult for all but the simplest tasks, since reward functions that depend on affecting change in the world to specific objects require perceiving those objects explicitly. **Thirdly**, we must be able to learn in a non-episodic manner, without access to episodic resets. A setup with explicit resets is increasingly impractical due to the requirement for significant human engineering or intervention during learning.

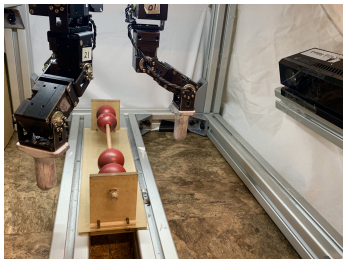

Figure 1: Our real world system setup. There is no special object instrumentation - only an RGB camera and proprioceptive sensors.

While some of the components discussed above can be tackled in isolation by current algorithms, there are unique challenges inherent to assembling these components into a complete learning system for real world robotics, as well as certain challenges associated with individual components. In the subsequent discussion, we outline the elements that are required to build a robotic system that can learn in the real world with minimal instrumentation. These elements present interesting challenges in learning when combined together, which we analyze in Section 3 and address in Section 4.

2.1 Learning from Raw Sensory Input

To enable learning without complex state estimation systems or instrumenting every environment the robot operates in, we require our robotic systems to be able to learn from their own raw sensory observations. Typically, these sensory observations are raw camera images from a camera mounted on the robot and proprioceptive sensory inputs such as the joint angles. These observations do not

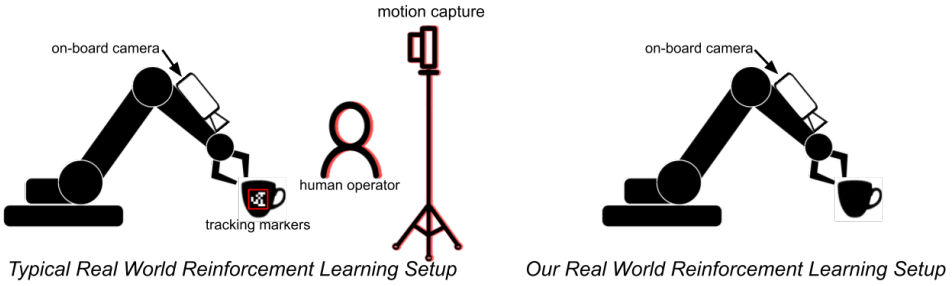

Figure 2: Schematic comparison of current learning systems versus our proposed instrumentation-free system - R3L . While traditional robotic applications of RL in the real world require explicit supervision in the form of resets, rewards and state estimation, R3L gets rid of these requirements and allows us to learn without any explicit system instrumentation, simply by leveraging interaction with the environment.

directly provide the poses of the objects in the scene, which is the typical assumption in simulated robotic environments – any such information must be extracted by the learning system.

While in principle many RL frameworks can support learning from raw sensory inputs (Mnih et al., 2015; Schulman et al., 2015; Lillicrap et al., 2015), it is important to consider the practicalities of this approach. For instance, we can instantiate vision-based RL with policy gradient algorithms such as TRPO (Schulman et al., 2015) and PPO (Schulman et al., 2017), but these have high sample complexities which make them unsuited for real world robotic learning (Haarnoja et al., 2018a), which is further exacerbated when learning from visual inputs. In our work, we consider adopting the general framework of off-policy actor-critic reinforcement learning, using a version of the soft actor critic (SAC) algorithm described by (Haarnoja et al., 2018b). This algorithm effectively uses off-policy data, and has been shown to learn some tasks directly from visual inputs. However, while SAC is able to learn directly from raw visual input, most instantiations have required instrumentation for providing rewards and episodic resets. As we show in Section 3, when these assumptions are lifted it leads to a number of non-trivial learning challenges which require new techniques to address.

2.2 Reward Functions without Reward Engineering

Vision-based RL algorithms, such as SAC, rely on a proper reward function being provided to the system, which is typically hand-defined by a user. While this can be simple to provide in simulation, it is significantly harder to implement in uninstrumented real world environments. In the real world, the robot must obtain *reward signal* itself from its own sensor readings, which can be extremely challenging. A few unappealing options have been suggested to tackle this: engineer complete computer vision systems to detect objects and extract reward signals (Devin et al., 2018; Nagabandi et al., 2019), engineer reward functions that use various task-specific heuristics to obtain rewards from pixels (Schenck & Fox, 2017; Kalashnikov et al., 2018), or instrument every environment (Chebotar et al., 2017). All are highly manual, tedious processes, and a better solution is needed to scale real world robotic learning gracefully.

To devise a system that requires minimal human engineering and supervision for providing rewards, we must use algorithms that are able to assign *themselves* rewards throughout learning with minimal reward engineering. One candidate is for a user to specify intended behavior beforehand through example images of desired goals. The algorithm can then assign itself reward based on a notion of how well it is accomplishing the specified goals during learning, with no additional human supervision. This scheme scales well since it requires minimal human engineering, and goal images are easy to provide upfront. So how exactly might we design such a reward provision system?

To do this, we use a data-driven reward specification framework called variational inverse control with events (VICE), introduced by Fu et al. (2018). VICE learns rewards in a task-agnostic way: we provide the algorithm with success examples in the form of images where the task is accomplished, and learn a discriminator that is capable of distinguishing successes from failures. This discriminator can then be used to provide a learning signal to nudge the reinforcement learning agent towards success. This algorithm has been previously considered in the context of learning some tasks from raw sensory observations in the real world by (Singh et al., 2019), but we show that it presents unique

challenges when used in conjunction with learning without episodic resets. Details and specifics of the algorithms being considered are described in Appendix A and also in (Fu et al., 2018; Singh et al., 2019).

2.3 Learning Without Resets

While the components described in Section 2.1 and 2.2 are essential to building continuously learning RL systems in the real world, they have often been implemented with the assumption of episodic learning. Indeed, previous applications of Fu et al. (2018) or Haarnoja et al. (2018b) were implemented with explicitly designed reset mechanisms or human operators performing resets between trials. However, natural open-world settings do not provide any such reset mechanism, and in order to enable scalable and autonomous real-world learning, we need systems that do not require an episodic formulation of the learning problem.

In principle, algorithms such as SAC do not actually require episodic learning; however, in practice, most instantiations of these algorithms have used explicit resets, even in simulation, and removing resets has resulted in failure to solve challenging tasks. While RL algorithms can handle non-episodic settings without any modifications in principle, they struggle when applied in practice to challenging problems. In our experiments in Section 3, we see that simply applying actor-critic methods to the reset free setting doesn't learn intended behaviors and requires novel insights when combined with visual observations and classifier based rewards.

These three components – vision-based RL with actor-critic algorithms, vision-based goal classifier for rewards, and reset-free learning – are the fundamental pieces that we need to build a real world robotic learning system. However, when we actually combine the individual components in Sections 3 and 6, we find that learning effective policies is quite challenging. We provide insight into these challenges in Section 3 and, based on these insights, we propose a number of simple but important changes in Section 4 that enable R3L to learn effectively and autonomously in the real world without human intervention.

3 Challenges of Real World RL

The system design outlined in Section 2, in principle, gives us a complete system to perform real world reinforcement learning without instrumentation. However, when ported to robotic problems in the real world, we find this basic design to be largely ineffective. To illustrate this, we present results for a simulated robotic manipulation task that requires repositioning a free-floating objects with a three-fingered robotic hand, shown in Fig 3. We use this task for our investigative analysis, and show that the same insights extend to several other tasks (including real world tasks) in Section 6. The goal in this task is to reposition the object to a target pose from any initial pose in the arena. When the system is instantiated with vision-based SAC, rewards from goal images using VICE and run without episodic resets, we see that the algorithm fails to make progress (Fig 4). Although it might appear that this setup fits within the assumptions of all of the components that are used, the complete system is ineffective. Which particular components of this problem make it so difficult?

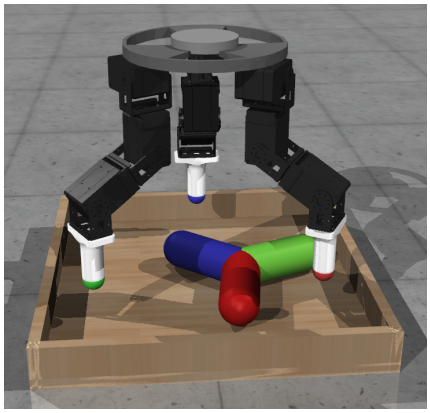

Figure 3: This is the object repositioning task. The goal is to move the object from any starting configuration to a particular goal position and orientation.

To investigate this issue, we set up experiments that combine the three main ingredients: varying observation type (visual vs. low-dimensional state), reward structure (VICE vs. hand-defined rewards that utilize ground-truth object state) and the ability to reset (episodic resets vs. reset-free, non-episodic learning). We start by considering the training time reward under each combination of factors as shown in Fig 4. This reveals several trends: first, the results in Fig 4 show that learning with resets gets

VICE / True Reward	With Resets	Without Resets
State	700k / 200k	1M / 500k
Vision	× / 800k	× / ×

Figure 4: We report the number of samples needed for the learned policy to achieve average training performance of less than 0.15 in pose distance (defined in App C.1.3) across 3 seeds. We compare training with true rewards vs. classifier-based rewards, with vs. without external resets, and from state vs. from vision on the object re-positioning task. We observe that learning without resets is more challenging than with resets and is much more challenging when combined with visual inputs.

Figure 5: We observe that when training reset free to reach a single goal, while the pose distance at training time is quite low, the pose errors obtained at test-time with the learned policy are very high. This indicates that while the object is getting close to the goal at training time, the policies being learned are still not effective.

good training time reward with both vision and state while reset-free only obtains good training time reward with low-dimensional state; second, we find that the policy is able to pass the threshold for training time rewards in a non-episodic setting when operating from low-dimensional state, but not when using image observations. This suggests that combining the reset-free learning problem with visual observations makes it significantly more challenging than with low-dimensional state.

However, the table in Fig 4 paints an incomplete picture. These numbers are related to the performance of the policies at training time, not how effective the learned policies are when being evaluated. When we consider the test-time performance (Fig 5) of the learned policies under reset free conditions, we obtain a different set of conclusions. While learning from low-dimensional state in the reset free setting is able to achieve decent training time reward, the test-time performance of the corresponding learned policies is very poor. This can likely be attributed to the fact that when the agent spends all its time stuck at the goal, it sees very little diversity of data in other parts of the state space, which significantly affects the efficacy of the actual policies being learned. This makes it very challenging to learn policies with completely reset-free schemes, which has prompted prior work to consider schemes such as learning reset-controllers (Eysenbach et al., 2018). As we discuss in the following section and in our experiments, these schemes are often insufficient for learning effective policies in real world without *any* resets. These insights prompt us to propose simple solutions for instrumentation free reinforcement learning in Section 4.

4 R3L: Real-world Robotic Reinforcement Learning

To address the challenges identified in Section 3, we present two improvements to the basic system outlined above, which we found to be essential for uninstrumented real-world training: randomized perturbation controllers and unsupervised representation learning. Incorporating these components into the system in Section 2 results in a method that can learn in uninstrumented environments, as we will show in Section 6.

4.1 Random Perturbation Controller

From our observations in Fig 4, we can see that it is not effective to simply perform reset-free RL using a standard actor critic algorithm. Even the policies trained with full state information do not actually learn how to perform the desired repositioning task, as shown in Fig 5. This is because, once the policy has performed the task once, it does not need to perform it again, and therefore not only fails how to learn to perform the task reliably, but in fact tends to forget how to perform it at all. Prior work has considered addressing this problem by converting the reset-free learning problem into a more standard episodic problem, by learning a "reset controller," which is trained to

reset the system to a particular *initial state* (Eysenbach et al., 2018; Han et al., 2015). However, as we will show in our experiments in Section 6, this still results in policies that only succeed from a very narrow range of initial states. Indeed, prior reset controller methods all reset to a *single* initial state (Eysenbach et al., 2018; Han et al., 2015).

We take a different approach to learning in a reset-free setting. Rather than attributing the problem to the variance of the initial state distribution, we hypothesize that a major problem with reset-free learning is that the support of the distribution of states visited by the policy is extremely narrow, which makes the learning problem challenging and doesn't allow the agent to learn how to perform the desired task from *any* state it might find itself in. In this view, the goal should not be to *reduce* the variance of the initial state distribution, but instead to *increase* it.

To this end, we utilize what we call random perturbation controllers: controllers that introduce perturbations intermittently into the system through a policy that is trained to explore the environment. The standard actor $\pi(a|s)$ is executed for H time-steps, following which we executed the perturbation controller $\pi_p(a|s)$ for H steps, and repeat. The policy π is trained with the VICE-based rewards for reaching the desired goals, while the perturbation controller π_p is trained only with an intrinsic motivation objective that encourages visiting under-explored states. In our implementation, we use the random network distillation (RND) objective for training the perturbation controller (Burda et al., 2018). This procedure is described in detail in Appendix A, and is evaluated on the tasks we consider in Fig 6. The perturbation controller ensures that the support of the training distribution grows, and as a result the policies can learn the desired behavior much more effectively, as shown in Fig 7.

4.2 Unsupervised Representation Learning

The perturbation controller discussed above allows us to learn policies that can succeed at the task from a variety of starting states. However, learning from visual observations still present a challenge. Our experiments in Fig 4 show that learning without resets from low-dimensional state is comparatively easier. We therefore aim to convert the vision-based learning problem into one that more closely resembles state-based learning, by training a variational autoencoder (VAE) and sharing the latent-variable representation across the actor and critic networks. Refer to Appendix B for more details. Note that we use a VAE as an instantiation of representation learning techniques that works well in the domains considered; approaches proposed by Lee et al. (2019), Hjelm et al. (2019) and Anand et al. (2019) are also viable substitutes in this regard.

While several prior works have also sought to incorporate unsupervised learning into reinforcement learning to make learning from images easier (Nair et al., 2018; Lee et al., 2019), we note that this becomes especially critical in the vision-based, reset-free setting, as motivated by the experiments in Section 3, which indicate that it is precisely this combination of factors – vision and no resets – that presents the most difficult learning problem. Therefore, although the particular solution we use in our system has been studied in prior work, it is brought to bare to address a challenge that arises in real-world learning that we believe has not been explored in prior studies.

These two improvements – the perturbation controller and joint training with an unsupervised learning loss, – combined with the general system described above, give us a complete practical system for real world reinforcement learning, which we term R3L . The overall method uses soft-actor critic for learning with visual observations and classifier based rewards with VICE, introduces auxiliary reconstruction objectives for unsupervised representation learning, and uses a perturbation controller during training to ensure that the learned policy can accomplish the task from a wide variety of states. Further details on the full system can be found in Appendix A.

5 Related Work

The primary contribution of this work is to propose a paradigm for continual instrumentation-free real world robotic learning, and a practical instantiations of such a system. Several prior works have applied policies learned with RL to particular tasks in the real world (Kalashnikov et al., 2018; Gu et al., 2017; Zhu et al., 2019; Nagabandi et al., 2019; Haarnoja et al., 2018b). While many of these algorithms simply train in simulation and transfer resulting policies to the real world, this paradigm is prone to domain shift and extensive simulation efforts (Sadeghi & Levine; Tobin et al., 2017; Andrychowicz et al., 2018). We instead focus on the paradigm of reinforcement learning

purely in the real world. While algorithms have shown that we can indeed perform RL in the real world (Kalashnikov et al., 2018; Gu et al., 2017; Zhu et al., 2019; Nagabandi et al., 2019; Haarnoja et al., 2018b; Kumar et al., 2016), these have been limited to highly instrumented laboratory settings. They have carefully hand-designed reward assignment schemes (Levine & Koltun, 2013), reset mechanisms (Gu et al., 2017; Chebotar et al., 2017; Zhu et al., 2019). In contrast, we are proposing to learn in environments with significantly less human instrumentation. This introduces a unique set of challenges not considered carefully in prior works.

A key component of our system is learning from raw visual inputs. This has proven to be extremely difficult for policy gradient style algorithms (Pinto et al., 2017a) due to challenging representation learning problems. This has been made easier in simulated domains by using modified objectives such as auxiliary losses (Jaderberg et al., 2016) or by using more efficient algorithms (Haarnoja et al., 2018a). We show that reinforcement learning on raw visual input, while possible in standard RL settings, becomes significantly more challenging when considered in conjunction with non-episodic, reset-free scenarios.

Reward function design is crucial for any RL system, and is non-trivial to provide in the real world. Prior works have considered instrumenting the environment with additional sensors to evaluate rewards (Gu et al., 2017; Chebotar et al., 2017; Zhu et al., 2019), which is a highly manual process, using demonstrations, which require manual effort to collect (Vecerik et al., 2017; Ng & Russell, 2000; Liu et al., 2018), or using interactive supervision from a user (Christiano et al., 2017). In this work, we leverage the algorithm introduced by Fu et al. (2018) to assign rewards based on the likelihood of a goal classifier. While prior work also applied this method to robotic tasks (Singh et al., 2019), this was done in a setting where manual resets were provided, while we demonstrate that we can use learned rewards in a fully uninstrumented, reset-free setup.

Learning without resets has been considered in prior works (Eysenbach et al., 2018; Han et al., 2015), although in different contexts – safe learning and learning compound controllers respectively. Eysenbach et al. (2018) provide an algorithm to learn a reset controller with the goal of ensuring safe operation, but makes several assumptions that make it difficult to use in the real world: it assumes access to a ground truth reward function, it assumes access to an oracle function that can detect if an attempted reset by the reset policy was successful or not, and it assumes the ability to perform manual resets if the reset policy fails a certain number of times. In contrast, we propose an algorithm that allows for fully automated reinforcement learning in the real world. We compare to an ablation of our method that uses a reset controller similar to Eysenbach et al. (2018), and show that our method performs substantially better. Our perturbation controller also resembles the adversarial RL setup Pinto et al. (2017b); Sukhbaatar et al. (2018). However, while these prior methods explicitly aim to train policies that are robust to perturbations Pinto et al. (2017b) or explore effectively Sukhbaatar et al. (2018), we are concerned with learning without access to resets.

While this line of work has connections to developmental robotics (Lungarella et al., 2003; Asada et al., 2009) and it's subfields such as continual (Lesort et al., 2019) and lifelong (Thrun & Mitchell, 1995) learning, the goal of our work is to handle the practicalities of enabling reinforcement learning systems to learn in the real world without instrumentation or interruption, even for a single task setting, without multi-task considerations. The insights we make should also be applicable to improve developmental robotics algorithms. Though these settings don't touch on all aspects of developmental robotics, our work does relate with the subfields of continual learning (Lesort et al., 2019), intrinsic motivation (Schmidhuber, 2006) and sensory-motor development involving proprioceptive manipulation (Stoica, 2001).

6 EXPERIMENTS

In our experimental evaluation, we study how well the R3L system, described in Sections 2 and 4, can learn under realistic settings – visual observations, no hand-specified rewards, and no resets. We consider the following hypotheses:

1. Can we use R3L to learn complex robotic manipulation tasks without instrumentation? Does this system learn skills in both simulation and the real world?
2. Do the solutions proposed in Section 4 actually enable R3L to perform tasks without instrumentation that would not have been otherwise possible?

6.1 EXPERIMENTAL SETUP

We consider the task of dexterous manipulation with a three fingered robotic hand, called the D'Claw (Zhu et al., 2019; Ahn et al., 2019), on a number of simulated and real world environments. These tasks involve complex coordination of three fingers with 3 DoF each in order to manipulate objects. Prior works that used this robot utilized explicit resets and low-dimensional true state observations, while we consider settings with visual observations, no hand-specified rewards, and no resets.

Figure 6: Visualizations of the goal configurations of the simulated and real world tasks being considered. From left to right we depict valve rotation, bead manipulation and free object repositioning in simulation, as well as valve rotation and bead manipulation manipulation in the real world.

The tasks in our experiments are shown in Fig 6: manipulating beads on an abacus row, valve rotation, and free object repositioning. These tasks represent a wide class of problems that robots might encounter in the real world. For each task, we consider the problem of reaching the depicted goal configuration: moving the abacus beads to a particular position, rotating the valve to a particular angle, and repositioning the free object to a particular goal position. For each task, policies are evaluated from a wide selection of initial configurations. Additional details about the tasks, and the evaluation procedures, including initial configurations and success metrics are provided in Appendix C. Videos and additional details can be found at `https://sites.google.com/view/realworld-rl/`.

6.2 LEARNING IN SIMULATION WITHOUT INSTRUMENTATION

We compare our entire proposed system implementation (Section 4) with a number of baselines and ablations. Importantly, all methods must operate under the same assumptions: no system instrumentation for state estimation, reward specification, or episodic resets. Firstly, we compare the performance of R3L to a system which uses SAC for vision-based RL from raw pixels, VICE for providing rewards and running reset-free (denoted as "VICE"). This corresponds to the vanilla version of R3L (Section 2), with none of the proposed insights and changes. We then compare with prior reset-free RL algorithms (Eysenbach et al., 2018) that explicitly learn a reset controller to alternate goals in the state space ("Reset Controller + VAE"). Lastly, we compare algorithm performance with two clear ablations - running R3L without the perturbation controller ("VICE + VAE") and without the unsupervised learning ("R3L w/o VAE"). This highlights the significance of all the components of R3L .

From the experimental results in Fig 7, it is clear that R3L is able to reach the best performance across tasks, while none of the other methods are able to solve all of the tasks. Different prior methods and ablations fail for different reasons: (1) methods without the reconstruction objective struggle at parsing the high-dimensional input and are unable to solve the harder task; (2) methods without the perturbation controller are ineffective at learning how to reach the goal from novel initialization positions for the more challenging object repositioning tasks, as discussed in Section 4.

We note that an explicit reset controller, which can be viewed as a softer version of our perturbation controller with goal-directedness, learns to solve the easier tasks due to the reset controller encouraging exploration of the state space. In our experiments for free object repositioning, performance was reported across 3 choices of reset states. The high variance in evaluation performance indicates that the performance of such a controller (or a goal conditioned variant of it) is highly sensitive to the choice of reset states. A poor choice of reset states, such as two that are very close together, may yield poor exploration leading to performance similar to the single goal VICE baseline. Furthermore, the choice of reset states is highly task dependent and it is often not clear what choice

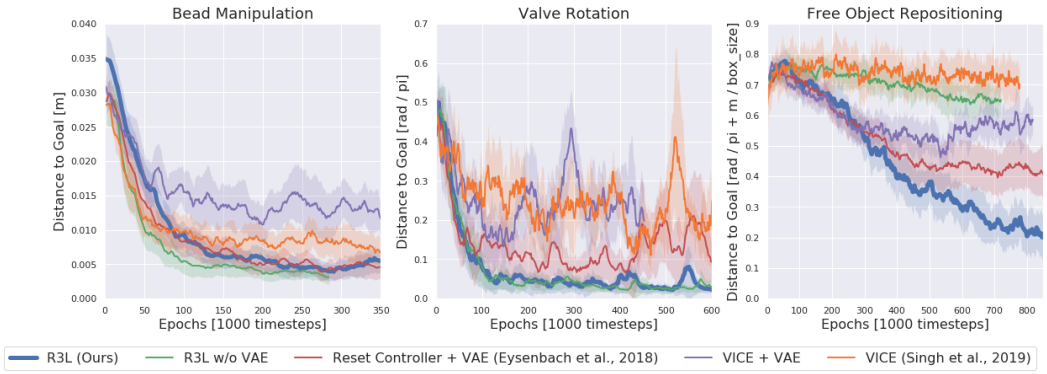

Figure 7: Quantitative evaluation of performance in simulation for bead manipulation, valve rotation and free object repositioning (left to right) run with 10, 5, 5 random seeds respectively. The error bars show 95% bootstrap confidence intervals for average performance. While other variants are sufficient to get good evaluation performance on easier tasks, harder tasks like free object repositioning require random perturbations and unsupervised representation learning to learn skills reset-free. See Appendix C.1 for details of evaluation procedures.

of goals will yield the best performance. On the contrary, our method does not require such task-specific knowledge and uses random perturbations to reset while training: this allows for a robust, instrumentation-free controller while also ensuring fast convergence.

6.3 Learning in the Real World without Instrumentation

Since the aim of R3L is to enable uninstrumented training in the real world, we next evaluate our method on a real-world robotic system, providing evidence that our insights generalize to the real world *without any instrumentation*. In the instrumentation-free setting, we leave the robot unattended with a set of goal images provided and the algorithm learns the desired behavior through interaction. The experiments are performed on the DClaw robotic hand with an RGB camera as the only sensory input. Intermediate policies are saved at regular intervals, and evaluations of all policies are performed after training has completed. For valve rotation, we declare an evaluation rollout a success if the final orientation is within within 15° of the goal. For bead manipulation, we declare success if all the beads are within 2cm of the goal state. Fig 8 compares the performance of our method without supervised learning ("R3L w/o VAE") in the real-world against a baseline that uses SAC for vision-based RL from raw pixels, VICE for providing rewards and running reset-free (denoted as "VICE"). We see that our method learns policies that evaluate from nearly all the initial configurations, whereas VICE fails to evaluate from most initial configurations. Fig 9 depicts sample evaluation rollouts of the policies learned using our method. For further details about real world experiments see Appendix C.2.

Figure 8: Quantitative evaluation of performance in real-world for valve rotation and bead manipulation. Policies trained with perturbation controllers can effectively evaluate after 17 and 5 hours of training respectively. For more fine-grained reporting of results see Figs 13-16.

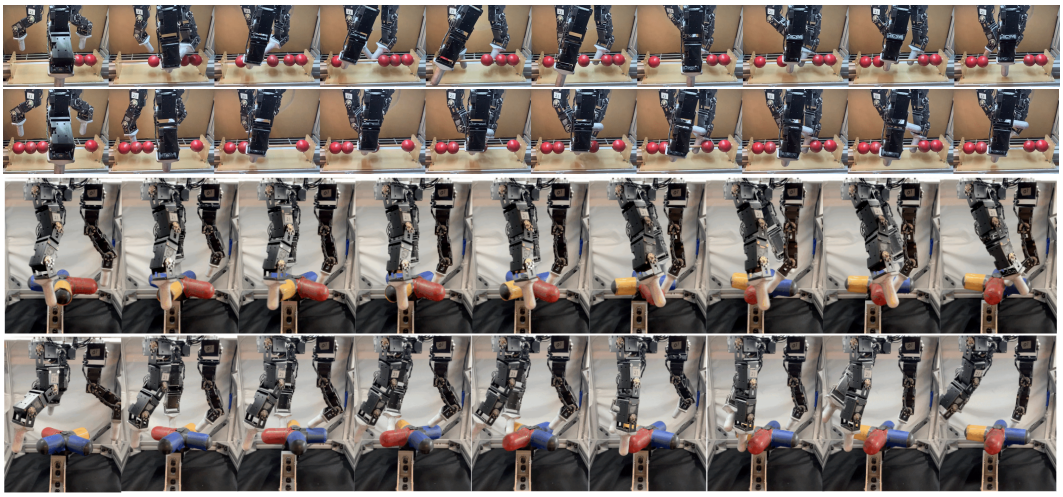

Figure 9: Evaluation rollouts of R3L on the real world tasks for policies trained without instrumentation. Successful evaluation rollouts for bead manipulation (top) and valve rotation (bottom) are shown here.

7 Conclusion

We presented the design and instantiation of R3L , a system for real world reinforcement learning. We identify and investigate the various ingredients required for such a system to scale gracefully with minimal human engineering and supervision. We show that this system must be able to learn from raw sensory observations, learn from very easily specified reward functions without reward engineering, and learn without any episodic resets. We describe the basic elements that are required to construct such a system, and identify unexpected learning challenges that arise from interplay of these elements. We propose simple and scalable fixes to these challenges through introducing unsupervised representation learning and a randomized perturbation controller. We show the effectiveness on such a system at learning without instrumentation in several simulated and real world environments.

The ability to train robots directly in the real world with minimal instrumentation opens a number of exciting avenues for future research. Robots that can learn unattended, without resets or hand-designed reward functions, can in principle collect very large amounts of experience autonomously, which may enable very broad generalization in the future. Furthermore, fully autonomous learning should make it possible for robots to acquire large behavioral repertoires, since each additional task requires only the initial examples needed to learn the reward. However, there are also a number of additional challenges, including sample complexity, optimization and exploration difficulties on more complex tasks, safe operation, communication latency, sensing and actuation noise, and so forth, all of which would need to be addressed in future work in order to enable truly scalable real-world robotic learning.

References

Michael Ahn, Henry Zhu, Kristian Hartikainen, Hugo Ponte, Abhishek Gupta, Sergey Levine, and Vikash Kumar. ROBEL: RObotics BEnchmarks for Learning with low-cost robots. In *Conference on Robot Learning (CoRL)*, 2019.

Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R. Devon Hjelm. Unsupervised state representation learning in atari. *arXiv*, 2019.

Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018.

Minoru Asada, Koh Hosoda, Yasuo Kuniyoshi, Hiroshi Ishiguro, Toshio Inui, Yuichiro Yoshikawa, Masaki Ogino, and Chisato Yoshida. Cognitive developmental robotics: A survey. *IEEE Trans.*

Autonomous Mental Development, 1(1):12–34, 2009. doi: 10.1109/TAMD.2009.2021702. URL https://doi.org/10.1109/TAMD.2009.2021702.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Yevgen Chebotar, Karol Hausman, Marvin Zhang, Gaurav Sukhatme, Stefan Schaal, and Sergey Levine. Combining model-based and model-free updates for trajectory-centric reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 703–711. JMLR. org, 2017.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pp. 4299–4307, 2017.

Coline Devin, Pieter Abbeel, Trevor Darrell, and Sergey Levine. Deep object-centric representations for generalizable robot learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7111–7118. IEEE, 2018.

Benjamin Eysenbach, Shixiang Gu, Julian Ibarz, and Sergey Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2018.

Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine. Variational inverse control with events: A general framework for data-driven reward definition. In *Advances in Neural Information Processing Systems*, 2018.

Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 3389–3396. IEEE, 2017.

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*, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Weiqiao Han, Sergey Levine, and Pieter Abbeel. Learning compound multi-step controllers under unknown dynamics. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6435–6442. IEEE, 2015.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bklr3j0cKX.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

Vikash Kumar, Emanuel Todorov, and Sergey Levine. Optimal control with learned local models: Application to dexterous manipulation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 378–383. IEEE, 2016.

Alex X. Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *CoRR*, abs/1907.00953, 2019.

Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz Rodríguez. Continual learning for robotics. *CoRR*, abs/1907.00182, 2019. URL http://arxiv.org/abs/1907.00182.

Sergey Levine and Vladlen Koltun. Guided policy search. In *Proceedings of The 30th International Conference on Machine Learning*, 2013.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1118–1125. IEEE, 2018.

Max Lungarella, Giorgio Metta, Rolf Pfeifer, and Giulio Sandini. Developmental robotics: a survey. *Connection Science*, 15(4):151–190, 2003. doi: 10.1080/09540090310001655110.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540), 2015.

Anusha Nagabandi, Kurt Konoglie, Sergey Levine, and Vikash Kumar. Deep Dynamics Models for Learning Dexterous Manipulation. In *Conference on Robot Learning (CoRL)*, 2019.

Ashvin Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *CoRR*, abs/1807.04742, 2018.

Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, 2000. ISBN 1-55860-707-2.

Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*, 2017a.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2817–2826, International Convention Centre, Sydney, Australia, 06–11 Aug 2017b. PMLR. URL http://proceedings.mlr.press/v70/pinto17a.html.

F Sadeghi and S Levine. Cad2rl: Real single-image flight without a single real image. arxiv 2016. *arXiv preprint arXiv:1611.04201*.

Connor Schenck and Dieter Fox. Visual closed-loop control for pouring liquids. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2629–2636. IEEE, 2017.

Jürgen Schmidhuber. Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. *Connect. Sci.*, 18(2):173–187, 2006. doi: 10.1080/09540090600768658. URL https://doi.org/10.1080/09540090600768658.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Avi Singh, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine. End-to-end robotic reinforcement learning without reward engineering. In *Robotics: Science and Systems (RSS)*, 2019.

Adrian Stoica. Robot fostering techniques for sensory-motor development of humanoid robots. *Robotics and Autonomous Systems*, 37(2-3):127–143, 2001. doi: 10.1016/S0921-8890(01)00154-3. URL https://doi.org/10.1016/S0921-8890(01)00154-3.

Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. In *6th International Conference on Learning Representations, ICLR 2018*, 2018.

Sebastian Thrun and Tom M. Mitchell. Lifelong robot learning. *Robotics and Autonomous Systems*, 15(1-2):25–46, 1995. doi: 10.1016/0921-8890(95)00004-Y. URL https://doi.org/10.1016/0921-8890(95)00004-Y.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pp. 23–30. IEEE, 2017.

Matej Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin A. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *CoRR*, abs/1707.08817, 2017. URL http://arxiv.org/abs/1707.08817.

Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

Henry Zhu, Abhishek Gupta, Aravind Rajeswaran, Sergey Levine, and Vikash Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 3651–3657. IEEE, 2019.

A ALGORITHM DETAILS

Algorithm 1 Real-World Robotic Reinforcement Learning (R3L)

procedure R3L
  $N \leftarrow$ number of training epochs
  $H \leftarrow$ trajectory length (horizon)
  $n_{\text{VICE}} \leftarrow$ number of VICE classifier training iterations per epoch
  Initialize forward and perturbing policies π_0, π_1
  Obtain goal states s_i^E and initialize as a goal pool $\mathcal{G}$
  Initialize RND target and predictor networks $f(s)$, $\hat{f}(s)$
  Initialize VICE reward classifier $r_{\text{VICE}}(s)$
  Initialize replay buffer $\mathcal{D}$
  Collect initial exploration data and add to $\mathcal{D}$
  for $i = 1$ to N **do**
    $k \leftarrow i \,\%\, 2$
    for $t = 1$ to H **do**
      Sample $a_t \sim \pi_k(s_t)$
      Sample $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$
      if k == 0 **then**
        $r_t(s_t) = c_{\text{VICE}} * r_{\text{VICE}}(s_t) + c_{\text{RND}} * r_{\text{RND}}(s_t)$
      else if k == 1 **then**
        $r_t(s_t) = r_{\text{RND}}(s_t)$
      end if
      Sample batch from $\mathcal{D}$
      Update π_k with batch according to SAC
      Update RND predictor network with batch
      Update running estimate of classifier and RND reward standard deviations
    end for
    Add experience to the replay buffer with $\mathcal{D} \leftarrow \mathcal{D} \cup \tau_i$
    Sample an equal number of goal examples from $\mathcal{G}$ and negative examples from $\mathcal{D}$
    for $t = 1$ to n_{VICE} **do**
      Train the VICE classifier on this batch with binary labels
    end for
  end for
end procedure

B TRAINING DETAILS

B.0.1 HYPERPARAMETERS

General	
Standard deviation update coefficient	**0.99**
Image Sizes	[(16, 16, 3), **(32, 32, 3)**, (64, 64, 3)]
SAC	
Learning Rate	**3e-4**
γ	**0.99**
Batch Size	**256**
Convnet Filters	[**(64, 64, 64)**, (16, 32, 64)]
Stride	**(2, 2)**
Kernel Sizes	**(3, 3)**
Pooling	[MaxPool2D, **None**]
Actor/Critic FC Layers	[**(512, 512)**, (256, 256, 256)]
VICE	
n_{VICE}	[1, **5**, 10]
Batch Size	**128**
Learning Rate	**1e-4**
Mixup α	**Uniform(0, 1)**
Convnet Filters	[**(64, 64, 64)**, (16, 32, 64)]
Stride	**(2, 2)**
Kernel Sizes	**(3, 3)**
Pooling	[MaxPool2D, **None**]
FC Layers	[**(512, 512)**, (256, 256, 256)]
RND	
Learning Rate	**3e-4**
Batch Size	**256**
Convnet Filters	**(16, 32, 64)**
Stride	**(2, 2)**
Kernel Sizes	**(3, 3)**
Pooling	[MaxPool2D, **None**]
FC Layers	[**(512, 512)**, (256, 256, 256)]
VAE	
Learning Rate	**1e-4**
Batch Size	**256**
Encoder (Convnet)	**(64, 64, 32)**
Latent Dimension	[8, **16, 32**, 64]
β	[1e-3, 0.1, **0.5**, 1, 10]
Stride	**(2, 2)**
Kernel Sizes	**(3, 3)**
Pooling	[MaxPool2D, **None**]

The ranges of values listed above represent the hyperparameters we searched over, and the bolded values are what we use in the Section 6 experiments.

B.0.2 VICE

We use a variant of VICE which defines the reward as the logits of the classifier, notably omitting the $-log(\pi(a|s))$ term. We also regularize our classifier with mixup (Zhang et al., 2018). We train all of our experiments using 200 goal images, which takes under an hour to collect in the real world for each task.

B.0.3 RANDOM NETWORK DISTILLATION (RND)

We found it important to normalize the predictor errors, just as (Burda et al., 2018) did.

B.0.4 VAE

We train a standard beta-VAE to maximize the evidence lower bound, given by:

$$\mathbb{E}_{z\sim q_\phi(z|x)} p_\theta(x|z) - \beta D_{KL}(q_\phi(z|x)||p_\theta(z))$$

To collect training data, we sampled random states in the observation space. In the real world, this sampling can be replaced with training an exploratory policy (i.e. using the RND reward as the policy's only objective). The learned weights of the encoder of the VAE are frozen, and the latent input is used to train the policy for reset-free RL.

C TASK DETAILS

C.1 SIMULATED TASKS

We evaluated our system across three tasks in simulation: bead manipulation, valve rotation, free object repositioning.

C.1.1 BEAD MANIPULATION

The bead manipulation task involves an abacus rod with four beads that can slide freely. The goal is to position two beads on each end from any initial configuration of beads. This can take the form of sliding one bead over (if three beads start on one side), two beads over (if all four beads start on one side), splitting beads apart (all four beads start in the middle), or some intermediate combination of those. The true reward is defined as the mean goal distance of all four beads. Policies are evaluated starting from the 8 initial configurations depicted in Fig 10. Evaluation performance reported in Section 6 for this task is defined as the final reward averaged across the 8 evaluation rollouts.

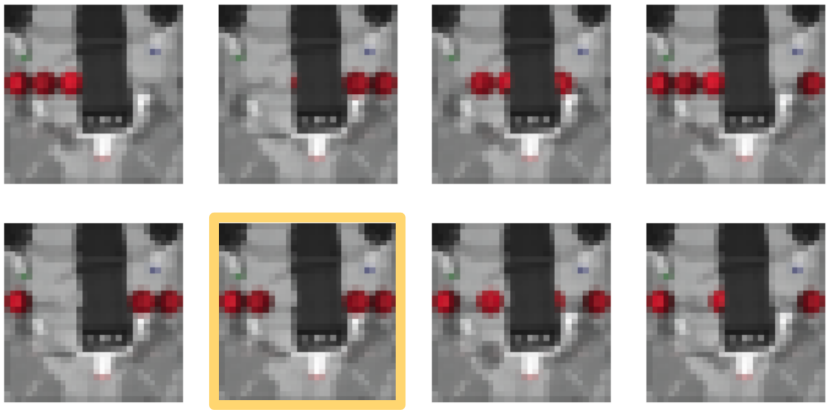

Figure 10: These are the 8 initial positions used for evaluating the performance of the bead manipulation policy. The goal configuration (which is also an initial evaluation position) is highlighted in yellow.

C.1.2 VALVE ROTATION

The claw is positioned above a three pronged valve (15 cm in diameter). The objective is to turn the valve to a given orientation from any initial orientation. The "true reward" is $r = -\log(|\theta_{state} - \theta_{goal}|)$. Policies are evaluated starting from the 8 initial configurations depicted in Fig 11. Evaluation performance reported in Section 6 for this task is defined as the final orientation distance averaged across the 8 evaluation rollouts.

C.1.3 FREE OBJECT REPOSITIONING

The claw is positioned atop a free (6 DoF) three pronged object (15cm in diameter), which can translate and rotate within a 30cm$x$30cm box. The goal is specified by a xy-position as well as a z-angle, where the xy-plane is the plane of the arena. The true reward is defined as the weighted sum of the angular and translational distances, $r = -2\log(||[x_{state}, y_{state}] - [x_{goal}, y_{goal}]||_2) -$

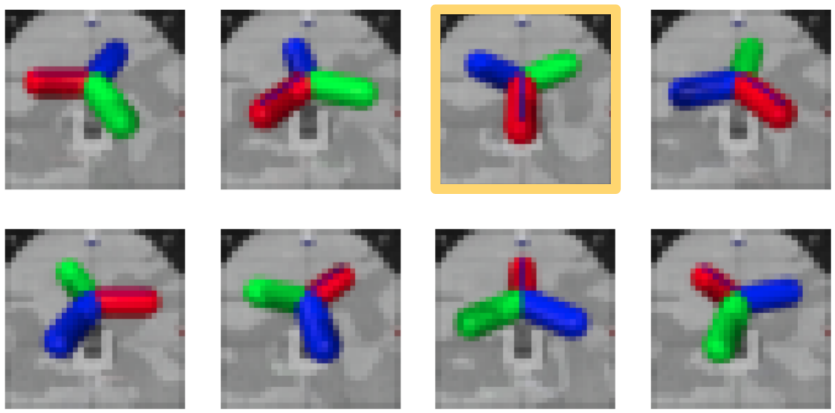

Figure 11: These are the 8 initial positions used for evaluating the performance of the valve rotation policy. The goal configuration (which is also an initial evaluation position) is highlighted in yellow.

$\log(|\theta_{state} - \theta_{goal}|)$. In our experiments, $(x, y, \theta)_{goal} = (0, 0, -\frac{\pi}{2})$, where the origin is centered in the arena. Policies are evaluated starting from the 15 initial configurations depicted in Figure 12. Evaluation performance reported in Section 6 for this task is defined as the final pose distance $(\frac{||[x_{final}, y_{final}] - [x_{goal}, y_{goal}]||_2}{0.25 \text{ m}} + \frac{|\theta_{final} - \theta_{goal}|}{\pi \text{ rad}})$ averaged across the 15 evaluation rollouts.

In our reset controller experiments, we averaged evaluation performance over three different choices of reset states, where the first reset state is always the goal:

1. $(x, y, \theta)_{reset,1} = (x, y, \theta)_{goal}, (x, y, \theta)_{reset,2} = (0.05, 0.05, \frac{\pi}{2})$
2. $(x, y, \theta)_{reset,1} = (x, y, \theta)_{goal}, (x, y, \theta)_{reset,2} = (0, 0, -\frac{\pi}{6})$
3. $(x, y, \theta)_{reset,1} = (x, y, \theta)_{goal}, (x, y, \theta)_{reset,2} = (-0.04, -0.04, -\frac{\pi}{2})$

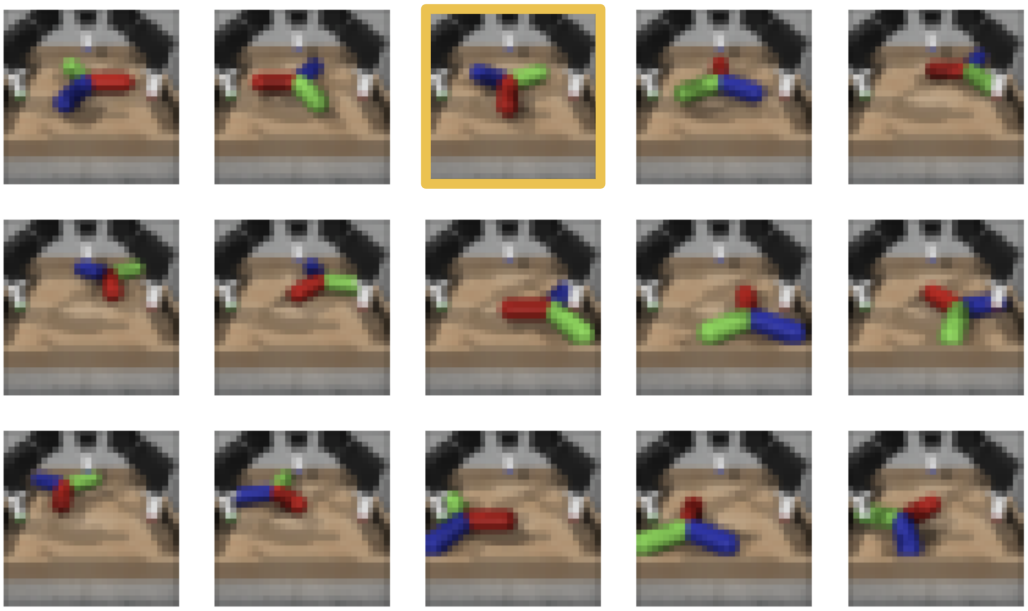

Figure 12: These are the 15 initial positions used for evaluating the performance of the free object repositioning policy. The goal configuration $(x, y, \theta)_{goal}$ which is also an initial evaluation position is highlighted in yellow.

C.2 REAL WORLD TASKS

For each setup we use an RGB camera to get images. We execute actions on the DClaw at 10Hz. In order to operate at such a high frequency while also training from images we sample and train asynchronously, but limit training to not exceed two gradient steps per transition sampled in the real world. Since direct performance metrics cannot be measured during training due to the lack of object instrumentation, evaluations of performance are done post-training.

C.2.1 VALVE ROTATION

The task is identical to the one in simulation. Evaluations were done post-training. An evaluation trajectory was defined as a success if at the last step, the valve was within 15 degrees of the goal. Each policy was evaluated over 8 rollouts, with initial configurations evenly spaced out between at increments of 45 degrees. Results are reported in Figures 13, 14.

C.2.2 BEAD MANIPULATION

The rod is 22cm in length, and each bead measures 3.5cm in diameter. Evaluations were done post-training, using the following procedure: the environment was manually reset to each of the 8 specified configurations shown in Figures 15 and 16 (which cover a full range of the state space) at the start of each evaluation rollout. An evaluation trajectory was defined as a success if at the last time step, all beads were within 2cm of their goal positions. Performance was evaluated at around 20 hours, at which point the policy achieved greater than 80% success on the 10 evaluation rollouts (a random policy achieved a success rate of 10%). Results are reported in Figs 15, 16.

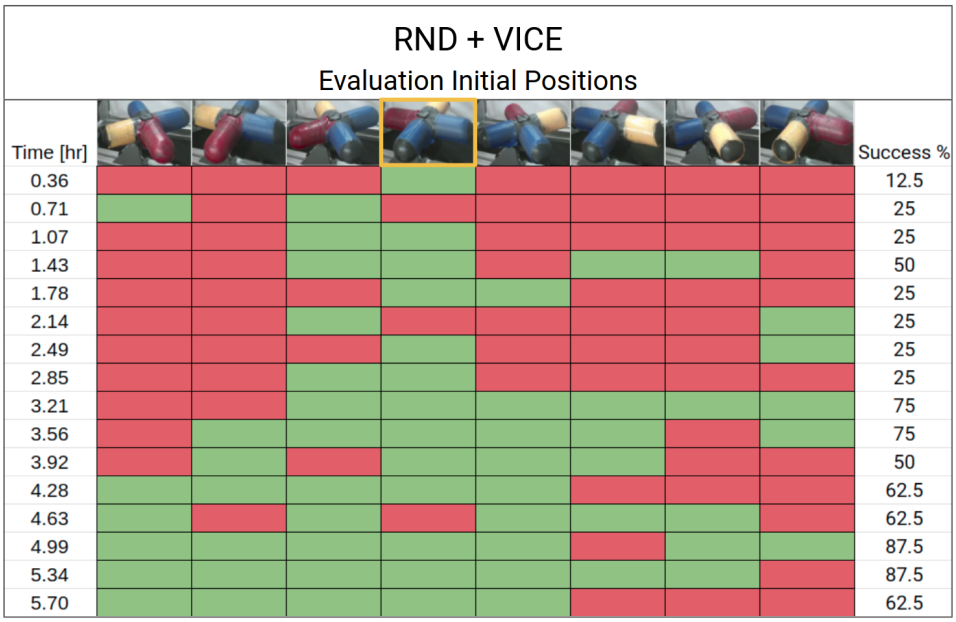

Figure 13: These are the results of the evaluation rollouts on the valve rotation task in the real world using our method (without the VAE). Trained policies were saved at regular intervals and evaluated post-training. Each row is a different policy, and each column an evaluation rollout from a different initial configuration. The goal is highlighted in yellow. Our method is able to achieve high success rates after 5 hours of training.

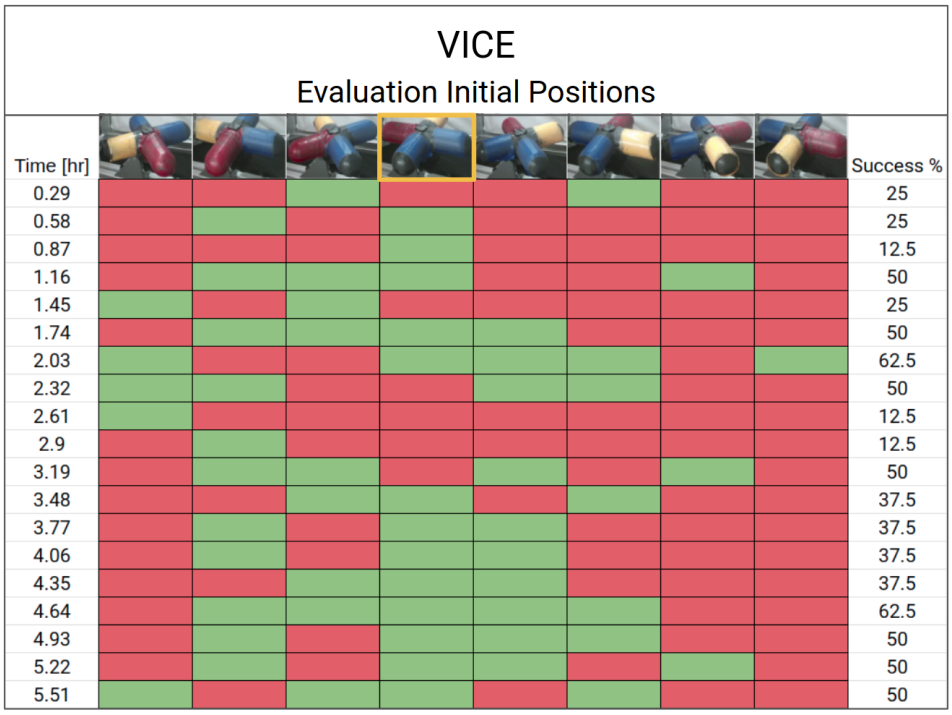

Figure 14: These are the results of evaluation rollouts on the valve rotation task in the real world using the VICE single goal baseline. The policies fail to evaluate well, especially from initial positions far from the goal position.

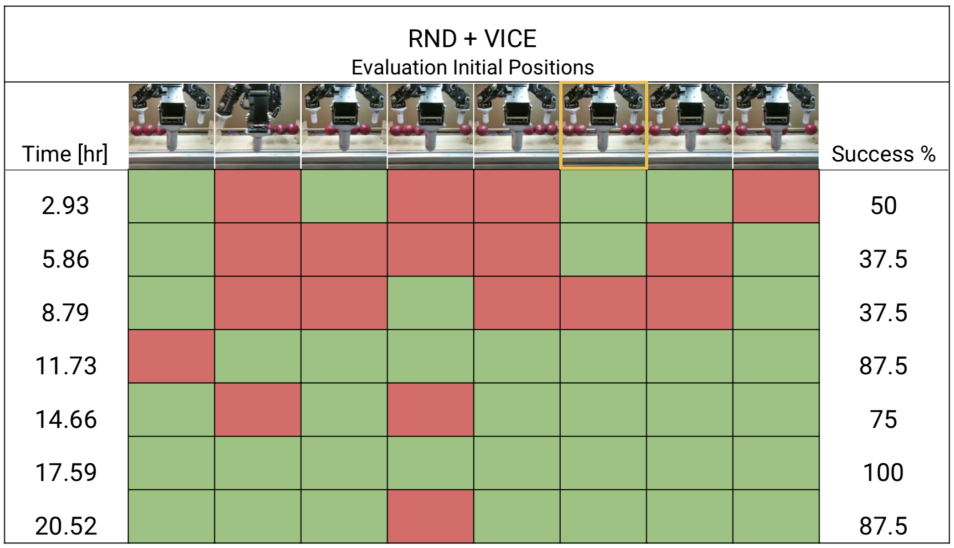

Figure 15: These are the results of the evaluation rollouts on the valve rotation task in the real world using our method (without the VAE). Trained policies were saved at regular intervals and evaluated post-training. Each row is a different policy, and each column an evaluation rollout from a different initial configuration. The goal is highlighted in yellow. Our method is able to achieve high success rates after 17 hours of training.

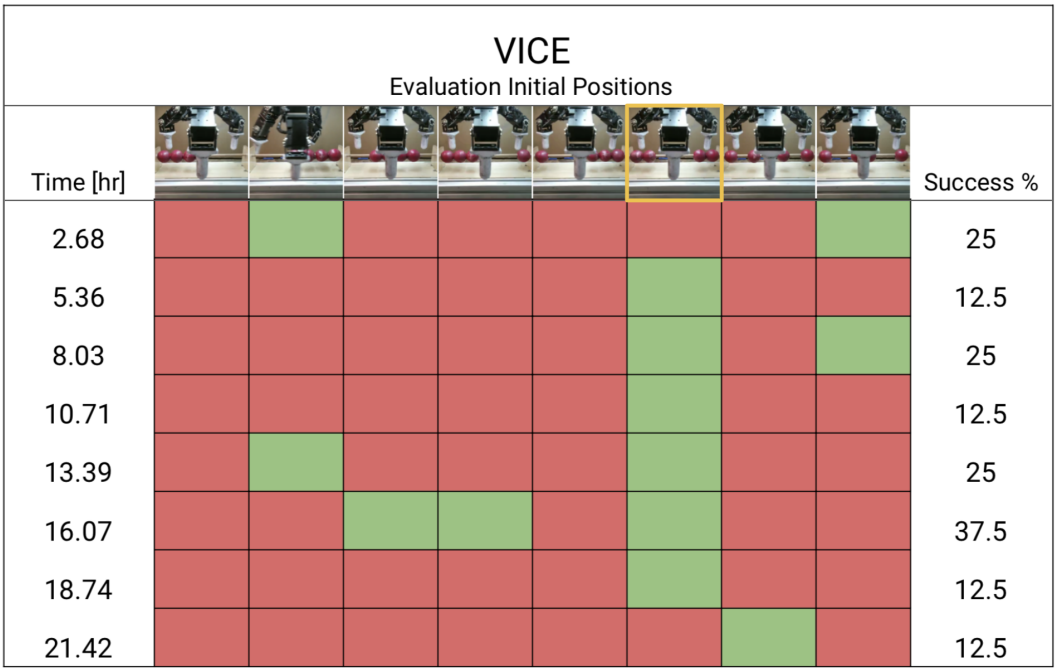

Figure 16: These are the results of evaluation rollouts on the valve rotation task in the real world using the VICE single goal baseline. The policies fail to evaluate consistently, except when the initial configuration matches the goal configuration.