

Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning

Tianhe Yu^{*1}, Deirdre Quillen^{*2}, Zhanpeng He^{*3}, Ryan Julian^{*4}, Avnish Narayan^{*4},
 Hayden Shively⁴, Adithya Bellathur⁴,
Karol Hausman⁵, Chelsea Finn¹, Sergey Levine²
 Stanford University¹, UC Berkeley², Columbia University³,
 University of Southern California⁴, Robotics at Google⁵

Abstract: Meta-reinforcement learning algorithms can enable robots to acquire new skills much more quickly, by leveraging prior experience to learn how to learn. However, much of the current research on meta-reinforcement learning focuses on task distributions that are very narrow. For example, a commonly used meta-reinforcement learning benchmark uses different running velocities for a simulated robot as different tasks. When policies are meta-trained on such narrow task distributions, they cannot possibly generalize to more quickly acquire entirely new tasks. Therefore, if the aim of these methods is enable faster acquisition of entirely new behaviors, we must evaluate them on task distributions that are sufficiently broad to enable generalization to new behaviors. In this paper, we propose an open-source simulated benchmark for meta-reinforcement learning and multi-task learning consisting of 50 distinct robotic manipulation tasks. Our aim is to make it possible to develop algorithms that generalize to accelerate the acquisition of entirely new, held-out tasks. We evaluate 7 state-of-the-art meta-reinforcement learning and multi-task learning algorithms on these tasks. Surprisingly, while each task and its variations (e.g., with different object positions) can be learned with reasonable success, these algorithms struggle to learn with multiple tasks at the same time, even with as few as ten distinct training tasks. Our analysis and open-source environments pave the way for future research in multi-task learning and meta-learning that can enable meaningful generalization, thereby unlocking the full potential of these methods.¹

Keywords: meta-learning, multi-task reinforcement learning, benchmarks

1 Introduction

While reinforcement learning (RL) has achieved some success in domains such as assembly [1], ping pong [2], in-hand manipulation [3], and hockey [4], state-of-the-art methods require substantially more experience than humans to acquire only one narrowly-defined skill. If we want robots to be broadly useful in realistic environments, we instead need algorithms that can learn a wide variety of skills reliably and efficiently. Fortunately, in most specific domains, such as robotic manipulation or locomotion, many individual tasks share common structure that can be reused to acquire related tasks more efficiently. For example, most robotic manipulation tasks involve grasping or moving objects in the workspace. However, while current methods can learn to individual skills like screwing on a bottle cap [1] and hanging a mug [5], we need algorithms that can efficiently learn shared structure across many related tasks, and use that structure to learn new skills quickly, such as screwing a jar lid or hanging a bag. Recent advances in machine learning have provided unparalleled generalization

* denotes equal contribution

¹Videos of the benchmark tasks are on the anonymous project page: meta-world.github.io.

Our open-sourced code for the benchmark is available at: <https://github.com/rlworkgroup/meta-world>.

All of the open-sourced baselines and launchers for our experiments can be found at <https://github.com/rlworkgroup/garage>.

This manuscript is an update on a manuscript that appeared at the 3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.

capabilities in domains such as images [6] and speech [7], suggesting that this should be possible; however, we have yet to see such generalization to diverse tasks in reinforcement learning settings.

Recent works in meta-learning and multi-task reinforcement learning have shown promise for addressing this gap. Multi-task RL methods aim to learn a single policy that can solve multiple tasks more efficiently than learning the tasks individually, while meta-learning methods train on many tasks, and optimize for fast adaptation to a new task. While these methods have made progress, the development of both classes of approaches has been limited by the lack of established benchmarks and evaluation protocols that reflect realistic use cases. On one hand, multi-task RL methods have largely been evaluated on disjoint and overly diverse tasks such as the Atari suite [8], where there is little efficiency to be gained by learning across games [9]. On the other hand, meta-RL methods have been evaluated on very narrow task distributions. For example, one popular evaluation of meta-learning involves choosing different running directions for simulated legged robots [10], which then enables fast adaptation to new directions. While these are technically distinct tasks, they are a far cry from the promise of a meta-learned model that can adapt to any new task within some domain. In order to study the capabilities of current multi-task and meta-reinforcement learning methods and make it feasible to design new algorithms that actually generalize and adapt quickly on meaningfully distinct tasks, we need evaluation protocols and task suites that are broad enough to enable this sort of generalization, while containing sufficient shared structure for generalization to be possible.

The key contributions of this work are a suite of 50 diverse simulated manipulation tasks and an extensive empirical evaluation of how previous methods perform on sets of such distinct tasks. We contend that multi-task and meta reinforcement learning methods that aim to efficiently learn many tasks and quickly generalize to new tasks should be evaluated on distributions of tasks that are diverse and exhibit shared structure. To this end, we present a benchmark of simulated manipulation tasks with everyday objects, all of which are contained in a shared, table-top environment with a simulated Sawyer arm. By providing a large set of distinct tasks that share common environment and control structure, we believe that this benchmark will allow researchers to test the generalization capabilities of the current multi-task and meta RL methods, and help to identify new research avenues to improve the current approaches. Our empirical evaluation of existing methods on this benchmark reveals that, despite some impressive progress in multi-task and meta-reinforcement learning over the past few years, current methods are generally not able to learn diverse task sets, much less generalize successfully to entirely new tasks. We provide an evaluation protocol with evaluation modes of varying difficulty, and observe that current methods show varying amounts of success on these modes. This opens the door for future developments in multi-task and meta reinforcement learning: instead of focusing on further increasing performance on current narrow task suites, we believe that it is essential for future work in these areas to focus on increasing the capabilities of algorithms to handle highly diverse task sets.

By doing so, we can enable meaningful generalization across many tasks and achieve the full potential of meta-learning as a means of incorporating past experience to make it possible for robots to acquire new skills as quickly as people can.

2 Related Work

Previous works that have proposed benchmarks for reinforcement learning have largely focused on single task learning settings [11, 12, 13]. One popular benchmark used to study multi-task learning is the Arcade Learning Environment, a suite of dozens of Atari 2600 games [14]. While having a tremendous impact on the multi-task reinforcement learning research community [9, 15, 8, 16, 17], the Atari games included in the benchmark have significant differences in visual appearance, controls, and objectives, making it challenging to acquire any efficiency gains through shared learning. In fact, many prior multi-task learning methods have observed substantial negative transfer between the Atari games [9, 15]. In contrast, we would like to study a case where positive transfer between the different tasks should be possible. We therefore propose a set of related yet diverse tasks that share the same robot, action space, and workspace.

Meta-reinforcement learning methods have been evaluated on a number of different problems, including maze navigation [18, 19, 20], continuous control domains with parametric variation across tasks [10, 21, 22, 23], bandit problems [19, 18, 20, 24], levels of an arcade game [25], and locomotion tasks with varying dynamics [26, 27]. Complementary to these evaluations, we aim to develop a testbed of tasks and an evaluation protocol that are reflective of the challenges in applying meta-

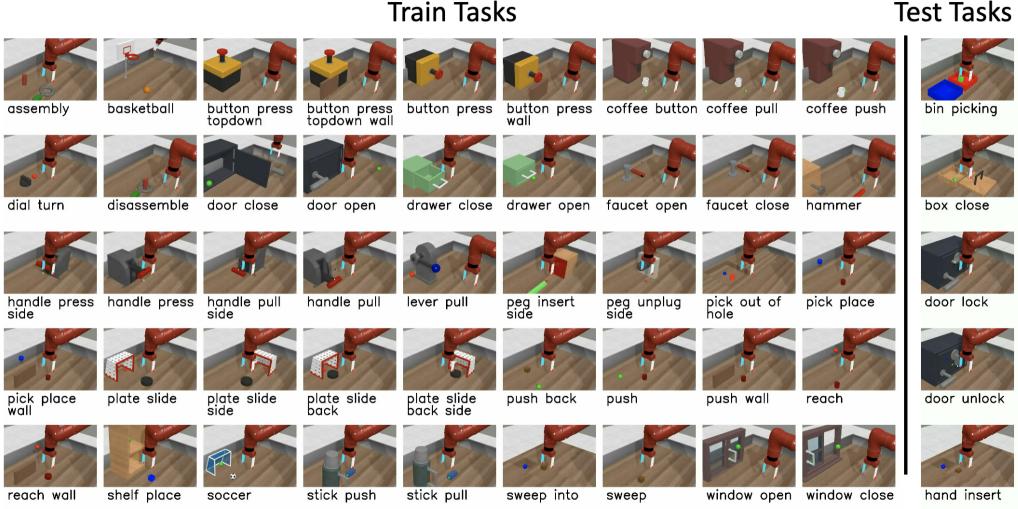


Figure 1: Meta-World contains 50 manipulation tasks, designed to be diverse yet carry shared structure that can be leveraged for efficient multi-task RL and transfer to new tasks via meta-RL. In the most difficult evaluation, the method must use experience from 45 training tasks (left) to quickly learn distinctly new test tasks (right). A larger view of the environments can be found on the next page.

learning to robotic manipulation problems, including both parametric and non-parametric variation in tasks.

There is a long history of robotics benchmarks [28, 29, 30], datasets [31, 32, 33, 34, 35, 36, 37], competitions [38] and standardized object sets [39, 40] that have played an important role in robotics research. Similarly, there exists a number of robotics simulation benchmarks including visual navigation [41, 42, 43, 44, 45], autonomous driving [46, 47, 48], grasping [49, 50, 51], single-task manipulation [52], among others. In this work, our aim is to continue this trend and provide a large suite of tasks that will allow researchers to study multi-task learning, meta-learning, and transfer in general. Further, unlike these prior simulation benchmarks, we particularly focus on providing a suite of many diverse manipulation tasks and a protocol for multi-task and meta RL evaluation.

3 The Multi-Task and Meta-RL Problem Statements

Our proposed benchmark is aimed at making it possible to study generalization in meta-RL and multi-task RL. In this section, we define the meta-RL and multi-task RL problem statements, and describe some of the challenges associated with task distributions in these settings.

We use the formalism of Markov decision processes (MDPs), where each task \mathcal{T} corresponds to a different finite horizon MDP, represented by a tuple (S, A, P, R, H, γ) , where $s \in S$ correspond to states, $a \in A$ correspond to the available actions, $P(s_{t+1}|s_t, a_t)$ represents the stochastic transition dynamics, $R(s, a)$ is a reward function, H is the horizon and γ is the discount factor. In standard reinforcement learning, the goal is to learn a policy $\pi(a|s)$ that maximizes the expected return, which is the sum of (discounted) rewards over all time. In multi-task and meta-RL settings, we assume a distribution of tasks $p(\mathcal{T})$. Different tasks may vary in any aspect of the Markov decision process, though efficiency gains in adaptation to new tasks are only possible if the tasks share some common structure. For example, as we describe in the next section, the tasks in our proposed benchmark have the same action space and horizon, and structurally similar rewards and state spaces.²

Multi-task RL problem statement. The goal of multi-task RL is to learn a single, task-conditioned policy $\pi(a|s, z)$, where z indicates an encoding of the task ID. This policy should maximize the average expected return across all tasks from the task distribution $p(\mathcal{T})$, given by $\mathbb{E}_{\mathcal{T} \sim p(\mathcal{T})}[\mathbb{E}_\pi[\sum_{t=0}^T \gamma^t R_t(s_t, a_t)]]$. The information about the task can be provided to the policy in various ways, e.g. using a one-hot task identification encoding z that is passed in addition to the cur-

²In practice, the policy must be able to read in the state for each of the tasks, which typically requires them to at least have the same dimensionality. In our benchmarks, some tasks have different numbers of objects, but the state dimensionality is always the same, meaning that some state coordinates are unused for some tasks.

rent state. There is no separate test set of tasks, and multi-task RL algorithms are typically evaluated on their average performance over the *training* tasks.

Meta-RL problem statement. Meta-reinforcement learning aims to leverage the set of training task to learn a policy $\pi(a|s)$ that can quickly adapt to new test tasks that were not seen during training, where both training and test tasks are assumed to be drawn from the same task distribution $p(\mathcal{T})$. Typically, the training tasks are referred to as the *meta-training* set, to distinguish from the adaptation (training) phase performed on the (meta-) test tasks. During meta-training, the learning algorithm has access to M tasks $\{\mathcal{T}_i\}_{i=1}^M$ that are drawn from the task distribution $p(\mathcal{T})$. At meta-test time, a new task $\mathcal{T}_j \sim p(\mathcal{T})$ is sampled that was not seen during meta-training, and the meta-trained policy must quickly adapt to this task to achieve the highest return with a small number of samples. A key premise in meta-RL is that a sufficiently powerful meta-RL method can meta-learn a model that effectively implements a highly efficient reinforcement learning procedure, which can then solve entirely new tasks very quickly – much more quickly than a conventional reinforcement learning algorithm learning from scratch. However, in order for this to happen, the meta-training distribution $p(\mathcal{T})$ must be sufficiently broad to encompass these new tasks. Unfortunately, most prior work in meta-RL evaluates on very narrow task distributions, with only one or two dimensions of parametric variation, such as the running direction for a simulated robot [10, 21, 22, 23].

4 Meta-World

If we want meta-RL methods to generalize effectively to entirely new tasks, we must meta-train on broad task distributions that are representative of the range of tasks that a particular agent might need to solve in the future. To this end, we propose a new multi-task and meta-RL benchmark, which we call Meta-World. In this section, we motivate the design decisions behind the Meta-World tasks, discuss the range of tasks, describe the representation of the actions, observations, and rewards, and present a set of evaluation protocols of varying difficulty for both meta-RL and multi-task RL.

4.1 The Space of Manipulation Tasks: Parametric and Non-Parametric Variability

A task, \mathcal{T} , in Meta-World is defined as the tuple (*reward function*, *initial object position*, *target position*). Meta-learning makes two critical assumptions: first, that the meta-training and meta-test tasks are drawn from the same distribution, $p(\mathcal{T})$, and second, that the task distribution $p(\mathcal{T})$ exhibits shared structure that can be utilized for efficient adaptation to new tasks. If $p(\mathcal{T})$ is defined as a family of variations within a particular control task, as in prior work [10, 22], then it is unreasonable to hope for generalization to entirely new control tasks. For example, an agent has little hope of being able to quickly learn to open a door, without having ever experienced doors before, if it has only been trained on a set of meta-training tasks that are homogeneous and narrow. Thus, to enable meta-RL methods to adapt to entirely new tasks, we propose a much larger suite of tasks consisting of 50 qualitatively-distinct manipulation tasks, where continuous parameter variation cannot be used to describe the differences between tasks.

With such non-parametric variation, however, there is the danger that tasks will not exhibit enough shared structure, or will lack the task overlap needed for the method to avoid memorizing each of the tasks. Motivated by this challenge, we design each task to include parametric variation in object and goal positions, as illustrated in Figure 2. Introducing this parametric variability not only creates a substantially larger (infinite) variety of tasks, but also makes it substantially more practical to expect that a meta-trained model will generalize to acquire entirely new tasks more quickly, since varying the positions provides for

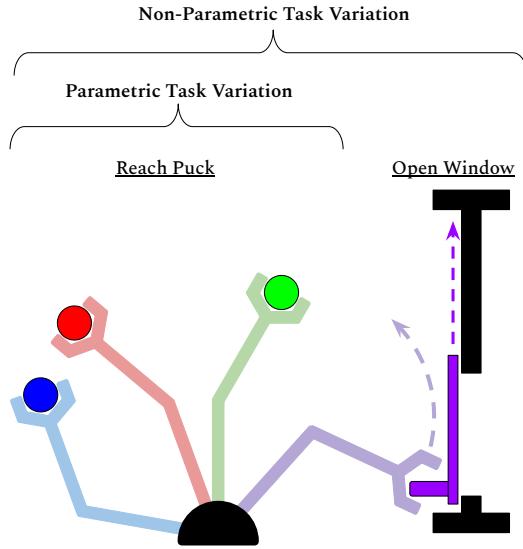


Figure 2: Parametric/non-parametric variation: all “reach puck” tasks (left) can be parameterized by the puck position, while the difference between “reach puck” and “open window” (right) is non-parametric.

wider coverage of the space of possible manipulation tasks. Without parametric variation, the model could for example memorize that any object at a particular location is a door, while any object at another location is a drawer. If the locations are not fixed, this kind of memorization is much less likely, and the model is forced to generalize more broadly. With enough tasks and variation within tasks, pairs of qualitatively-distinct tasks are more likely to overlap, serving as a catalyst for generalization. For example, closing a drawer and pushing a block can appear as nearly the same task for some initial and goal positions of each object.

Note that this kind of parametric variation, which we introduce *for each task*, essentially represents the entirety of the task distribution for previous meta-RL evaluations [10, 22], which test on single tasks (e.g., running towards a goal) with parametric variability (e.g., variation in the goal position). Our full task distribution is therefore substantially broader, since it includes this parametric variability *for each of the 50 tasks*.

To provide shared structure, the 50 environments require the same robotic arm to interact with different objects, with different shapes, joints, and connectivity. The tasks themselves require the robot to execute a combination of reaching, pushing, and grasping, depending on the task. By recombining these basic behavioral building blocks with a variety of objects with different shapes and articulation properties, we can create a wide range of manipulation tasks. For example, the **open door** task involves pushing or grasping an object with a revolute joint, while the **open drawer** task requires pushing or grasping an object with a sliding joint. More complex tasks require a combination of these building blocks, which must be executed in the right order. We visualize all of the tasks in Meta-World in Figure 1, and include a description of all tasks in Appendix A.

All of the tasks are implemented in the MuJoCo physics engine [53], which enables fast simulation of physical contact. To make the interface simple and accessible, we base our suite on the Multiworld interface [54] and the OpenAI Gym environment interfaces [11], making additions and adaptations of the suite relatively easy for researchers already familiar with Gym.

4.2 Actions, Observations, and Rewards

In order to represent policies for multiple tasks with one model, the observation and action spaces must contain significant shared structure across tasks. All of our tasks are performed by a simulated Sawyer robot. The action space is a 2-tuple consisting of the change in 3D space of the end-effector followed by a normalized torque that the gripper fingers should apply. The actions in this space range between -1 and 1 . For all tasks, the robot must either manipulate one object with a variable goal position, or manipulate two objects with a fixed goal position. The observation space is represented as a 6-tuple of the 3D Cartesian positions of the end-effector, a normalized measurement of how open the gripper is, the 3D position of the first object, the quaternion of the first object, the 3D position of the second object, the quaternion of the second object, all of the previous measurements in the environment, and finally the 3D position of the goal. If there is no second object or the goal is not meant to be included in the observation, then the quantities corresponding to them are zeroed out. The observation space is always 39 dimensional.

Designing reward functions for Meta-World requires two major considerations. First, to guarantee that our tasks are within the reach of current single-task reinforcement learning algorithms, which is a prerequisite for evaluating multi-task and meta-RL algorithms, we design well-shaped reward functions for each task that make each of the tasks at least individually solvable. More importantly, the reward functions must exhibit shared structure across tasks. Critically, even if the reward function admits the same optimal policy for multiple tasks, varying reward scales or structures can make the tasks appear completely distinct for the learning algorithm, masking their shared structure and leading to preferences for tasks with high-magnitude rewards [8]. Accordingly, we adopt a structured, multi-component reward function for all tasks, which leads to effective policy learning for each of the task components. For instance, in a task that involves a combination of reaching, grasping, and placing an object, let $o \in \mathbb{R}^3$ be the object position, where $o = (o_x, o_y, o_z)$, $h \in \mathbb{R}^3$ be the position of the robot’s gripper, $z_{\text{target}} \in \mathbb{R}$ be the target height of lifting the object, and $g \in \mathbb{R}^3$ be goal position. With the above definition, the multi-component reward function R is the combination of a reaching reward, a grasping reward, and a placing reward or subsets thereof for simpler tasks that only involve reaching and/or pushing. With this design, the reward functions across all tasks have a similar magnitude that ranges between 0 and 10, where 10 always corresponds to the reward-function being solved, and conform to similar structure, as desired. The full form of the reward function and a list of all task rewards is provided in Appendix E.

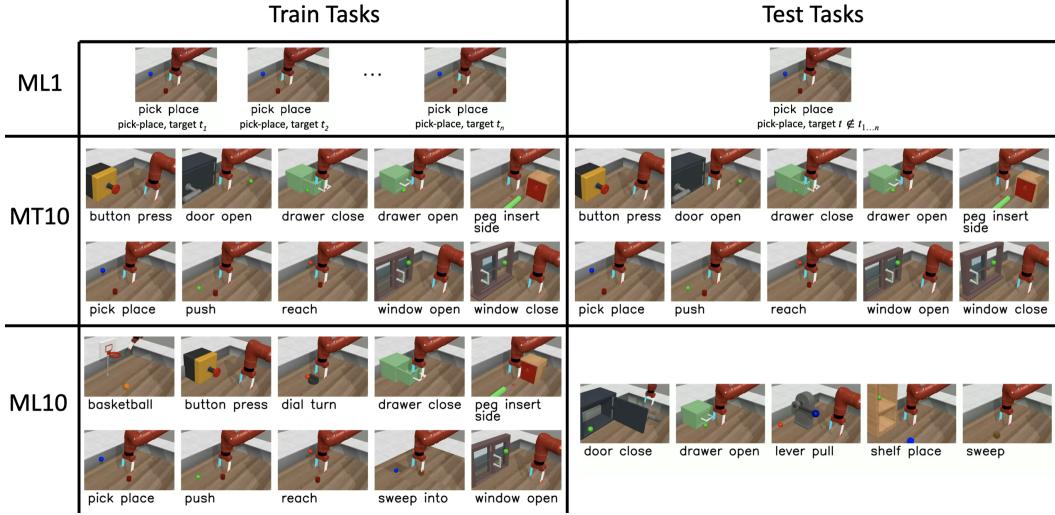


Figure 3: Visualization of three of our multi-task and meta-learning evaluation protocols, ranging from within task adaptation in ML1, to multi-task training across 10 distinct task families in MT10, to adapting to new tasks in ML10. Our most challenging evaluation mode ML45 is shown in Figure 1.

4.3 Evaluation Protocol

With the goal of providing a challenging benchmark to facilitate progress in multi-task RL and meta-RL, we design an evaluation protocol with varying levels of difficulty, ranging from the level of current goal-centric meta-RL benchmarks to a setting where methods must learn distinctly new, challenging manipulation tasks based on diverse experience across 45 tasks. We hence divide our evaluation into five categories, which we describe next. We then detail our evaluation criteria.

Meta-Learning 1 (ML1): Few-shot adaptation to goal variation within one task. The simplest evaluation aims to verify that previous meta-RL algorithms can adapt to new object or goal configurations on only one type of task. ML1 uses single Meta-World Tasks, with the meta-training “tasks” corresponding to 50 random initial object and goal positions, and meta-testing on 50 held-out positions. This resembles the evaluations in prior works [10, 22]. We evaluate algorithms on three individual tasks from Meta-World: reaching, pushing, and pick and place, where the variation is over reaching position or goal object position. The goal positions are not provided in the observation, forcing meta-RL algorithms to adapt to the goal through trial-and-error.

Multi-Task 1 (MT1): Learning one multi-task policy that generalizes to 50 tasks belonging to the same environment. This evaluation aims to verify how well multi-task algorithms can learn across a large related task distribution. MT1 uses single Meta-World environments, with the training “tasks” corresponding to 50 random initial object and goal positions. The goal positions are provided in the observation and are a fixed set, as to focus on the ability of algorithms in acquiring a distinct skill across multiple goals, rather than generalization and robustness.

Multi-Task 10, Multi-Task 50 (MT10, MT50): Learning one multi-task policy that generalizes to 50 tasks belonging to 10 and 50 training environments, for a total of 500, and 2,500 training tasks. A first step towards adapting quickly to distinctly new tasks is the ability to train a single policy that can solve multiple distinct training tasks. The multi-task evaluation in Meta-World tests the ability to learn multiple tasks at once, without accounting for generalization to new tasks. The MT10 evaluation uses 10 environments: reach, push, pick and place, open door, open drawer, close drawer, press button top-down, insert peg side, open window, and open box. The larger MT50 evaluation uses all 50 Meta-World environments. In our experiments, the algorithm is typically provided with a one-hot vector indicating the current task. The positions of objects and goal positions are fixed in all tasks in this evaluation, so as to focus on acquiring the distinct skills, rather than generalization and robustness.

Meta-Learning 10, Meta-Learning 45 (ML10, ML45): Few-shot adaptation to new test tasks with 10 and 50 meta-training tasks. With the objective to test generalization to new tasks, we hold

out 5 tasks and meta-train policies on 10 and 45 tasks. We randomize object and goals positions and intentionally select training tasks with structural similarity to the test tasks. Task IDs are not provided as input, requiring a meta-RL algorithm to identify the tasks from experience.

Success metrics. Since values of reward are not directly indicative how successful a policy is, we define an interpretable success metric for each task, which will be used as the evaluation criterion for all of the above evaluation settings. Since all of our tasks involve manipulating one or more objects into a goal configuration, this success metric is typically based on the distance between the task-relevant object and its final goal pose, i.e. $\|o - g\|_2 < \epsilon$, where ϵ is a small distance threshold such as 5 cm. For the complete list of success metrics and thresholds for each task, see Appendix 12.

5 Experimental Results and Analysis

The first, most basic goal of our experiments is to verify that each of the 50 presented tasks are indeed solvable by existing single-task reinforcement learning algorithms. We provide this verification in Appendix B. Beyond verifying the individual tasks, the goals of our experiments are to study the following questions: (1) can existing state-of-the-art meta-learning algorithms quickly learn qualitatively new tasks when meta-trained on a sufficiently broad, yet structured task distribution, and (2) how do different multi-task and meta-learning algorithms compare in this setting? To answer these questions, we evaluate various multi-task and meta-learning algorithms on the Meta-World benchmark. We include the training curves of all evaluations in Figure 15 in the Appendix C. Videos of the tasks and evaluations, along with all source code, are on the project webpage³.

In the multi-task evaluation, we evaluate the following RL algorithms: **multi-task proximal policy optimization (PPO)** [55]: a policy gradient algorithm adapted to the multi-task setting by providing the one-hot task ID as input, **multi-task trust region policy optimization (TRPO)** [56]: an on-policy policy gradient algorithm adapted to the multi-task setting using the one-hot task ID as input, **multi-task soft actor-critic (SAC)** [57]: an off-policy actor-critic algorithm adapted to the multi-task setting using the one-hot task ID as input, and an on-policy version of **task embeddings (TE)** [58]: a multi-task reinforcement learning algorithm that parameterizes the learned policies via shared skill embedding space. For the meta-RL evaluation, we study three algorithms: **RL²** [18, 19]: an on-policy meta-RL algorithm that corresponds to training a GRU network with hidden states maintained across episodes within a task and trained with PPO, **model-agnostic meta-learning (MAML)** [10, 21]: an on-policy gradient-based meta-RL algorithm that embeds policy gradient steps into the meta-optimization, and is trained with PPO, and **probabilistic embeddings for actor-critic RL (PEARL)** [22]: an off-policy actor-critic meta-RL algorithm, which learns to encode experience into a probabilistic embedding of the task that is fed to the actor and the critic. We use the baselines in the Garage [59] reinforcement learning library, which we developed for benchmarking Meta-World.

We show results of the simplest meta-learning evaluation mode, ML1, in Figure 4. We find that there is room for improvement even in this very simple setting. Next, we look at results of multi-task learning across distinct tasks, starting with MT10 in Figure 5 and in Table 1.

We find that multi-task SAC is able to learn the MT10 task suite well, achieving around 68% success rate averaged across tasks, while multi-task PPO and TRPO are only able to achieve around a 30% success rate. However, as we scale to 50 distinct tasks with MT50, we find that MT-SAC and MT-PPO only achieve around a 35-38% success rate, indicating that there is significant room for improvement in these methods.

Finally, we study the ML10 and ML45 meta-learning benchmarks, which require learning the meta-training tasks and generalizing to new meta-test tasks with small amounts of experience. From Figure 8 and Table 1, we find that the prior meta-RL methods, MAML and RL² reach 35% and 31% success on ML10 test tasks, while PEARL achieves only 13% on ML10. On ML45, MAML and RL² solve around 39.9% and 33.3% of the meta-test tasks. Note that, on both ML10 and ML45, the meta-training performance of all methods also has considerable room for improvement, suggesting that optimization challenges are generally more severe in the meta-learning setting. The fact that some methods nonetheless exhibit meaningful generalization suggests that the ML10 and ML45 benchmarks are solvable, but challenging for current methods, leaving considerable room for improvement in future work.

³Videos are on the project webpage, at meta-world.github.io

ML-1 Maximum Success Rates (N=10)

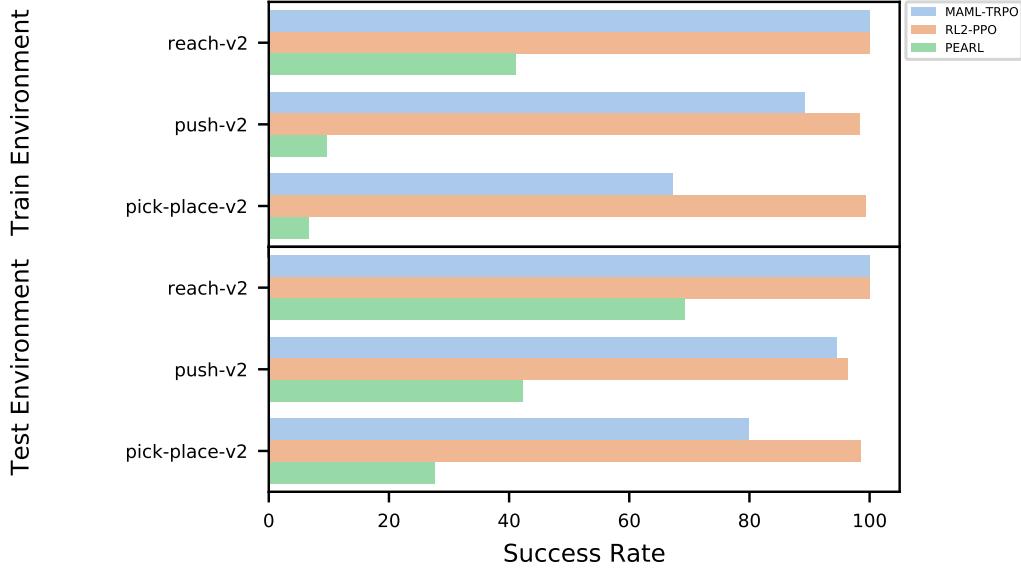


Figure 4: Comparison on our simplest meta-RL evaluation, ML1 on 10 seeds. RL^2 shows the strongest performance in generalization. Pearl shows the weakest performance, though this could be attributed to difficulty in training its task encoder

MT-10 Maximum Per-Task Success Rates (N=10)

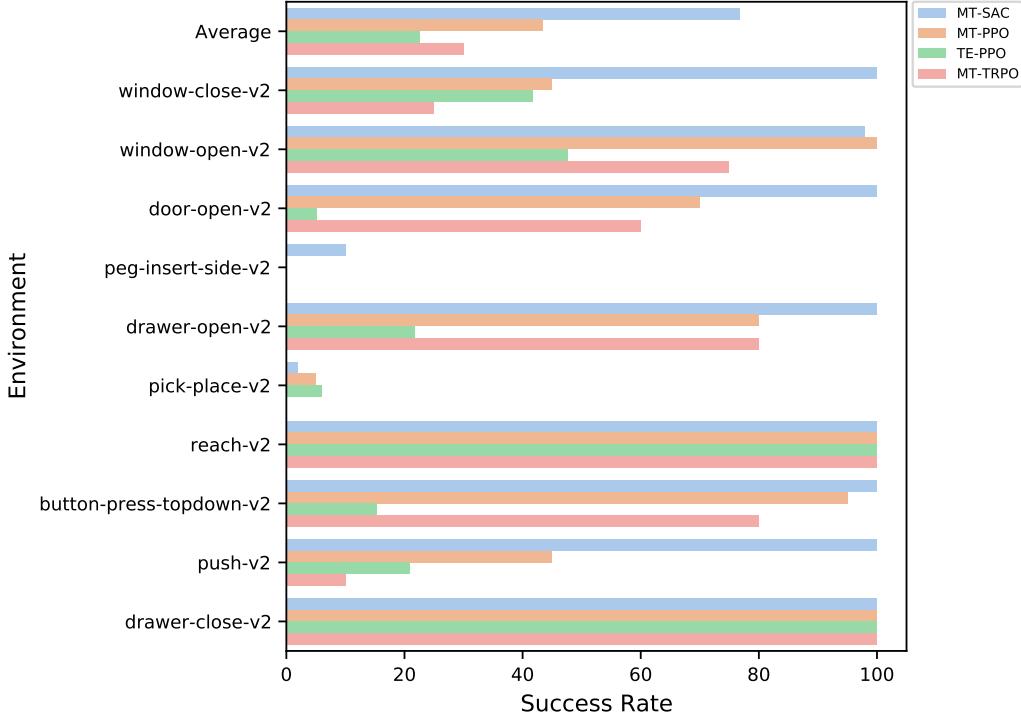


Figure 5: Performance of the tested MTRL algorithms on 10 seeds. MT-SAC performs the best on MT-10, exhibiting the greatest sample efficiency and performance. For detailed plots of these algorithm's learning curves, see appendix C.

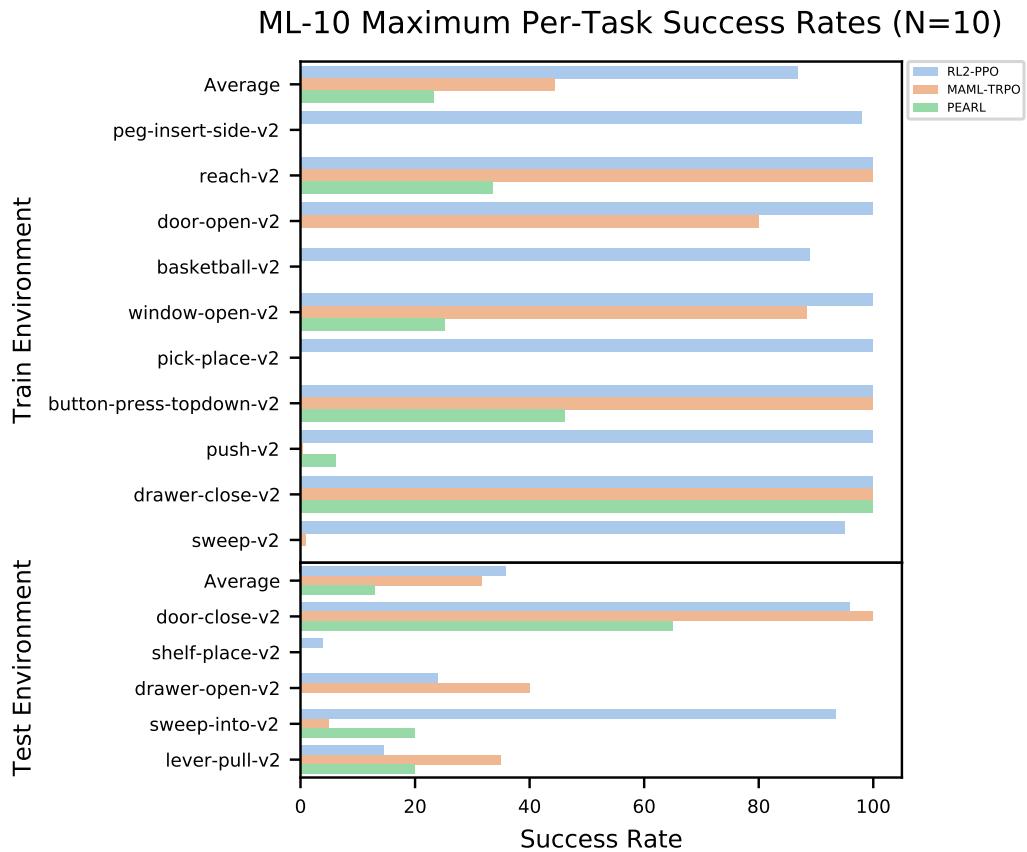


Figure 6: Performance of the tested meta-RL algorithms on 10 seeds. RL^2 shows the highest performance on the training tasks (86.9%), however its ability to generalize is not that much greater than MAML (35.8% for RL^2 and 31.6% for MAML).

MT-50 Maximum Per-Task Success Rates (N=10)

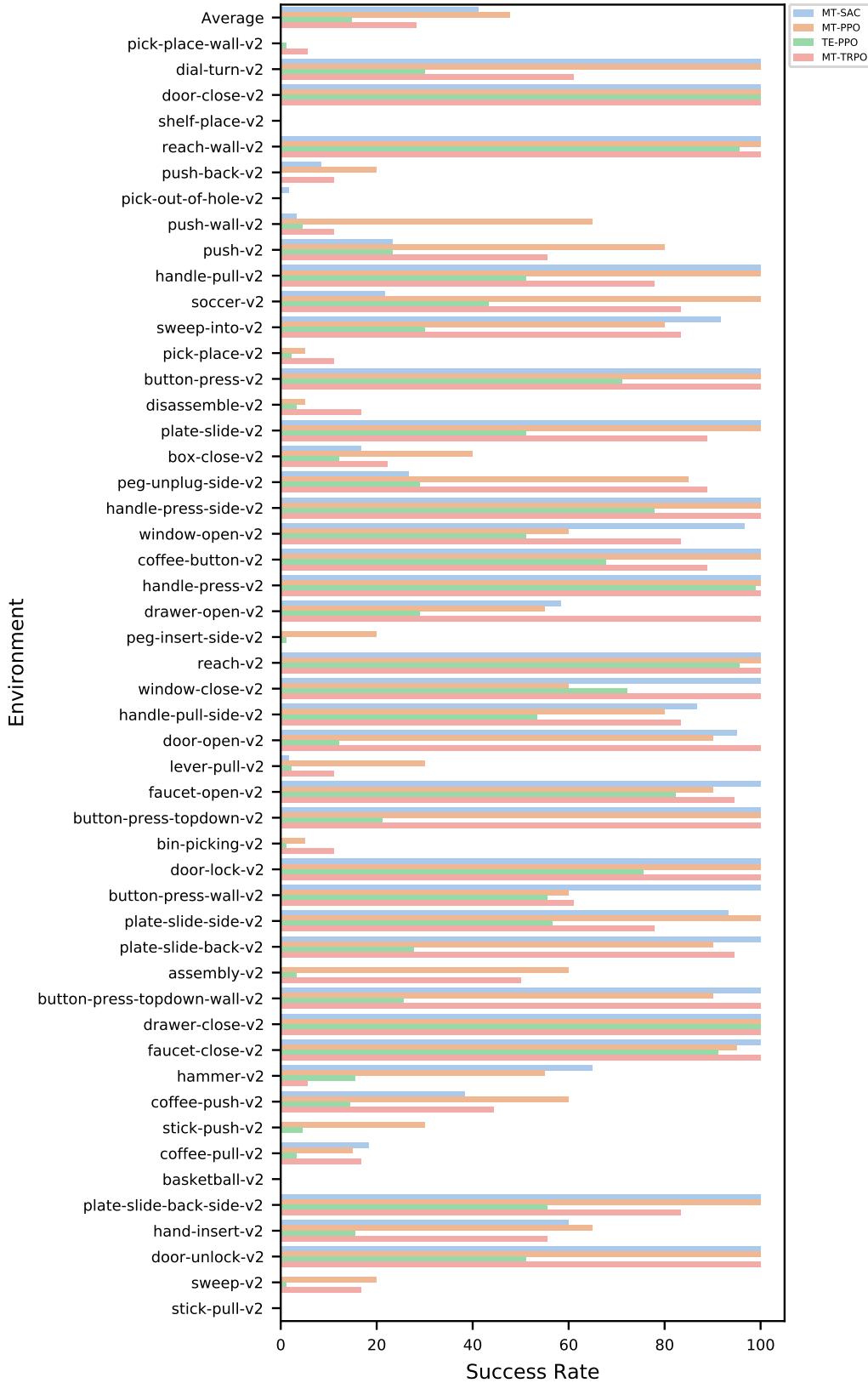


Figure 7: Performance of the tested MTRL algorithms on 10 seeds. In MT-10, MT-SAC showed the highest performance, however its performance does not scale to MT-50, the more difficult benchmark. MT-PPO exhibits the better performance in this benchmark.

ML-45 Maximum Per-Task Success Rates (N=10)

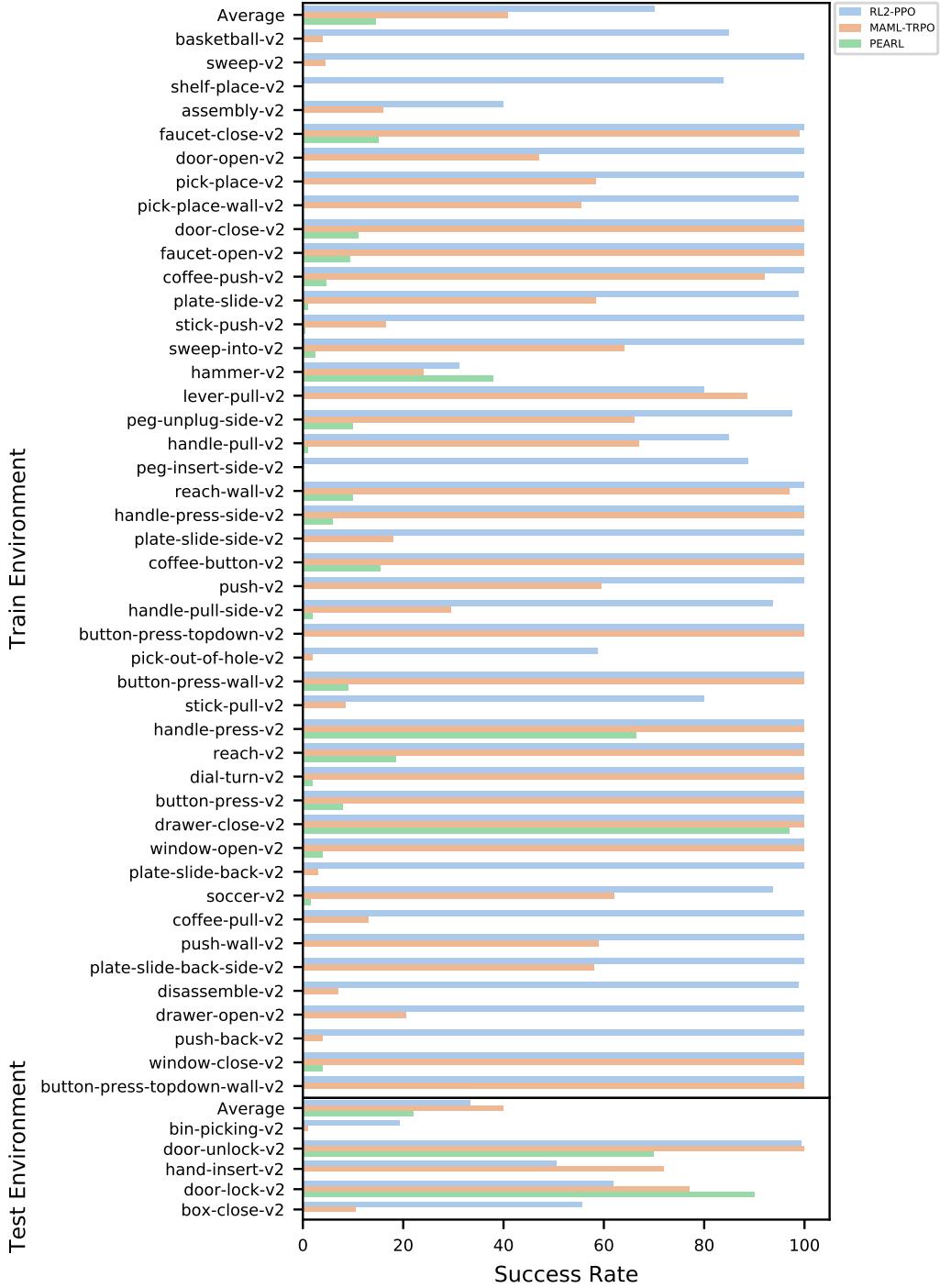


Figure 8: Average of maximum success rate for ML-45. Note that, even on the challenging ML-45 benchmark, current methods already exhibit some degree of generalization, but meta-training performance leaves considerable room for improvement, suggesting that future work could attain better performance on these benchmarks. Though PEARL has weak training performance, it has comparable performance on test tasks. RL² has the highest We also show the max average success rates for all benchmarks in Table 1.

Methods	MT10	MT50	ML10		ML45	
			meta-train	meta-test	meta-train	meta-test
Multi-task PPO	30.5%	35.4%	MAML	44.4%	31.6%	40.7%
Multi-task TRPO	31.3%	21.0%	RL ²	86.9%	35.8%	70%
Task embeddings	20.9%	11.8%	PEARL	23.2%	13%	33.3%
Multi-task SAC	68.3%	38.5%			14.5%	22%

Table 1: The average maximum success rate over all tasks for MT10, MT50, ML10, and ML45 on 10 seeds. The best performance in each benchmark is bolden. For MT10 and MT50, we show the average training success rate of multi-task SAC and multi-task PPO respectively outperform other methods. For ML10 and ML45, we show the meta-train and meta-test success rates. RL² achieves best meta-train performance in ML10 and ML45, while MAML and RL2 get the best generalization performance in ML10 and ML45 meta-test tasks respectively.

6 Conclusion and Directions for Future Work

We proposed an open-source benchmark for meta-reinforcement learning and multi-task learning, which consists of a large number of simulated robotic manipulation tasks.

Unlike previous evaluation benchmarks in meta-RL, our benchmark specifically emphasizes generalization to distinctly new tasks, not just in terms of parametric variation in goals, but completely new objects and interaction scenarios.

While meta-RL can in principle make it feasible for agents to acquire new skills more quickly by leveraging past experience, previous evaluation benchmarks utilize very narrow task distributions, making it difficult to understand the degree to which meta-RL actually enables this kind of generalization. The aim of our benchmark is to make it possible to develop new meta-RL algorithms that actually exhibit this sort of generalization. Our experiments show that current meta-RL methods in fact cannot yet generalize effectively to entirely new tasks and do not even learn the meta-training tasks effectively when meta-trained across multiple distinct tasks. This suggests a number of directions for future work, which we describe below.

Future directions for algorithm design. The main conclusion from our experimental evaluation with our proposed benchmark is that current meta-RL algorithms generally struggle in settings where the meta-training tasks are highly diverse. This issue mirrors the challenges observed in multi-task RL, which is also challenging with our task suite, and has been observed to require considerable additional algorithmic development to attain good results in prior work [9, 15, 16]. A number of recent works have studied algorithmic improvements in the area of multi-task reinforcement learning, as well as potential explanations for the difficulty of RL in the multi-task setting [8, 60]. Incorporating some of these methods into meta-RL, as well as developing new techniques to enable meta-RL algorithms to train on broader task distributions, would be a promising direction for future work to enable meta-RL methods to generalize effectively across diverse tasks, and our proposed benchmark suite can provide future algorithms development with a useful gauge of progress towards the eventual goal of broad task generalization.

Future extensions of the benchmark. While the presented benchmark is significantly broader and more challenging than existing evaluations of meta-reinforcement learning algorithms, there are a number of extensions to the benchmark that would continue to improve and expand upon its applicability to realistic robotics tasks. First, in many situations, the poses of objects are not directly accessible to a robot in the real world. Hence, one interesting and important direction for future work is to consider image observations and sparse rewards. Sparse rewards can be derived already using the success metrics, while support for image rendering is already supported by the code. However, for meta-learning algorithms, special care needs to be taken to ensure that the task cannot be inferred directly from the image, else meta-learning algorithms will memorize the training tasks rather than learning to adapt. Another natural extension would be to consider including a breadth of compositional long-horizon tasks, where there exist combinatorial numbers of tasks. Such tasks would be a straightforward extension, and provide the possibility to include many more tasks with shared structure. Another challenge when deploying robot learning and meta-learning algorithms is the manual effort of resetting the environment. To simulate this case, one simple extension of the benchmark is to significantly reduce the frequency of resets available to the robot while learning. Lastly, in many real-world situations, the tasks are not available all at once. To reflect this challenge in the benchmark, we can add an evaluation protocol that matches that of online meta-learning problem statements [61]. We leave these directions for future work, either to be done by ourselves

or in the form of open-source contributions. To summarize, we believe that the proposed form of the task suite represents a significant step towards evaluating multi-task and meta-learning algorithms on diverse robotic manipulation problems that will pave the way for future research in these areas.

Acknowledgments

We thank Suraj Nair for feedback on a draft of the paper. We thank K.R Zentner for her help in maintaining Meta-World. This research was supported in part by the National Science Foundation under IIS-1651843, IIS-1700697, and IIS-1700696, the Office of Naval Research, ARL DCIST CRA W911NF-17-2-0181, DARPA, Google, Amazon, and NVIDIA.

References

- [1] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research (JMLR)*, 2016.
- [2] K. Mülling, J. Kober, O. Kroemer, and J. Peters. Learning to select and generalize striking movements in robot table tennis. *IJRR*, 2013.
- [3] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. *arXiv:1808.00177*, 2018.
- [4] Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine. Combining model-based and model-free updates for trajectory-centric reinforcement learning. In *ICML*, 2017.
- [5] L. Manuelli, W. Gao, P. R. Florence, and R. Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. *CoRR*, abs/1903.06684, 2019.
- [6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012.
- [7] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.
- [8] M. Hessel, H. Soyer, L. Espeholt, W. Czarnecki, S. Schmitt, and H. van Hasselt. Multi-task deep reinforcement learning with popart. *CoRR*, abs/1809.04474, 2018.
- [9] E. Parisotto, J. L. Ba, and R. Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv:1511.06342*, 2015.
- [10] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.
- [11] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv:1606.01540*, 2016.
- [12] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman. Quantifying generalization in reinforcement learning. *arXiv:1812.02341*, 2018.
- [13] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, et al. Deepmind control suite. *arXiv:1801.00690*, 2018.
- [14] M. C. Machado, M. G. Bellemare, E. Talvitie, J. Veness, M. J. Hausknecht, and M. Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *CoRR*, abs/1709.06009, 2017.
- [15] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell. Policy distillation. *arXiv:1511.06295*, 2015.
- [16] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv:1802.01561*, 2018.
- [17] S. Sharma and B. Ravindran. Online multi-task learning using active sampling. 2017.
- [18] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning. *CoRR*, abs/1611.02779, 2016.
- [19] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick. Learning to reinforcement learn, 2016. *arXiv:1611.05763*.
- [20] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. A simple neural attentive meta-learner. *arXiv:1707.03141*, 2017.
- [21] J. Rothfuss, D. Lee, I. Clavera, T. Asfour, and P. Abbeel. Promp: Proximal meta-policy search. *arXiv:1810.06784*, 2018.
- [22] K. Rakelly, A. Zhou, D. Quillen, C. Finn, and S. Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables. *arXiv:1903.08254*, 2019.

- [23] C. Fernando, J. Sygnowski, S. Osindero, J. Wang, T. Schaul, D. Teplyashin, P. Sprechmann, A. Pritzel, and A. Rusu. Meta-learning by the baldwin effect. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1313–1320. ACM, 2018.
- [24] S. Ritter, J. X. Wang, Z. Kurth-Nelson, S. M. Jayakumar, C. Blundell, R. Pascanu, and M. Botvinick. Been there, done that: Meta-learning with episodic recall. *arXiv preprint arXiv:1805.09692*, 2018.
- [25] A. Nichol, V. Pfau, C. Hesse, O. Klimov, and J. Schulman. Gotta learn fast: A new benchmark for generalization in rl. *arXiv:1804.03720*, 2018.
- [26] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv:1803.11347*, 2018.
- [27] S. Sæmundsson, K. Hofmann, and M. P. Deisenroth. Meta reinforcement learning with latent variable gaussian processes. *arXiv:1803.07551*, 2018.
- [28] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar. Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols. *arXiv:1502.03143*, 2015.
- [29] Y. Lee, E. S. Hu, Z. Yang, A. Yin, and J. J. Lim. IKEA furniture assembly environment for long-horizon complex manipulation tasks. *CoRR*, abs/1911.07246, 2019. URL <http://arxiv.org/abs/1911.07246>.
- [30] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison. Rlbench: The robot learning benchmark and learning environment, 2019.
- [31] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *IJRR*, 2015.
- [32] 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.
- [33] K.-T. Yu, M. Bauza, N. Fazeli, and A. Rodriguez. More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing. In *IROS*, 2016.
- [34] Y. Chebotar, K. Hausman, Z. Su, A. Molchanov, O. Kroemer, G. Sukhatme, and S. Schaal. Bigs: Biotac grasp stability dataset. In *ICRA 2016 Workshop on Grasping and Manipulation Datasets*, 2016.
- [35] A. Gupta, A. Murali, D. P. Gandhi, and L. Pinto. Robot learning in homes: Improving generalization and reducing dataset bias. In *Advances in Neural Information Processing Systems*, pages 9112–9122, 2018.
- [36] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. *arXiv:1811.02790*, 2018.
- [37] P. Sharma, L. Mohan, L. Pinto, and A. Gupta. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. *arXiv preprint arXiv:1810.07121*, 2018.
- [38] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, and P. R. Wurman. Analysis and observations from the first amazon picking challenge. *IEEE Transactions on Automation Science and Engineering*, 15(1):172–188, 2016.
- [39] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *International Conference on Advanced Robotics (ICAR)*, 2015.
- [40] Y. S. Choi, T. Deyle, T. Chen, J. D. Glass, and C. C. Kemp. A list of household objects for robotic retrieval prioritized by people with als. In *International Conference on Rehabilitation Robotics*, 2009.
- [41] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, et al. Habitat: A platform for embodied ai research. *arXiv:1904.01201*, 2019.
- [42] E. Kolve, R. Mottaghi, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv:1712.05474*, 2017.
- [43] S. Brodeur, E. Perez, A. Anand, F. Golemo, L. Celotti, F. Strub, J. Rouat, H. Larochelle, and A. Courville. Home: A household multimodal environment. *arXiv:1711.11017*, 2017.
- [44] M. Savva, A. X. Chang, A. Dosovitskiy, T. Funkhouser, and V. Koltun. Minos: Multimodal indoor simulator for navigation in complex environments. *arXiv:1712.03931*, 2017.
- [45] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese. Gibson env: Real-world perception for embodied agents. In *Computer Vision and Pattern Recognition*, 2018.
- [46] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator. *arXiv:1711.03938*, 2017.

- [47] B. Wyman, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner. Torcs, the open racing car simulator. *Software available at <http://torcs.sourceforge.net>*, 4(6), 2000.
- [48] S. R. Richter, Z. Hayder, and V. Koltun. Playing for benchmarks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2213–2222, 2017.
- [49] D. Kappler, J. Bohg, and S. Schaal. Leveraging big data for grasp planning. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4304–4311. IEEE, 2015.
- [50] A. Kasper, Z. Xue, and R. Dillmann. The kit object models database: An object model database for object recognition, localization and manipulation in service robotics. *IJRR*, 2012.
- [51] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen. The columbia grasp database. 2008.
- [52] L. Fan, Y. Zhu, J. Zhu, Z. Liu, O. Zeng, A. Gupta, J. Creus-Costa, S. Savarese, and L. Fei-Fei. Surreal: Open-source reinforcement learning framework and robot manipulation benchmark. In *Conference on Robot Learning*, 2018.
- [53] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems*, 2012.
- [54] A. V. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, 2018.
- [55] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [56] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.
- [57] 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*, 2018.
- [58] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller. Learning an embedding space for transferable robot skills. *International Conference on Learning Representations*, 2018.
- [59] T. garage contributors. Garage: A toolkit for reproducible reinforcement learning research. <https://github.com/rlworkgroup/garage>, 2021.
- [60] T. Schaul, D. Borsa, J. Modayil, and R. Pascanu. Ray interference: a source of plateaus in deep reinforcement learning. *arXiv preprint arXiv:1904.11455*, 2019.
- [61] C. Finn, A. Rajeswaran, S. Kakade, and S. Levine. Online meta-learning. *ICML*, 2019.

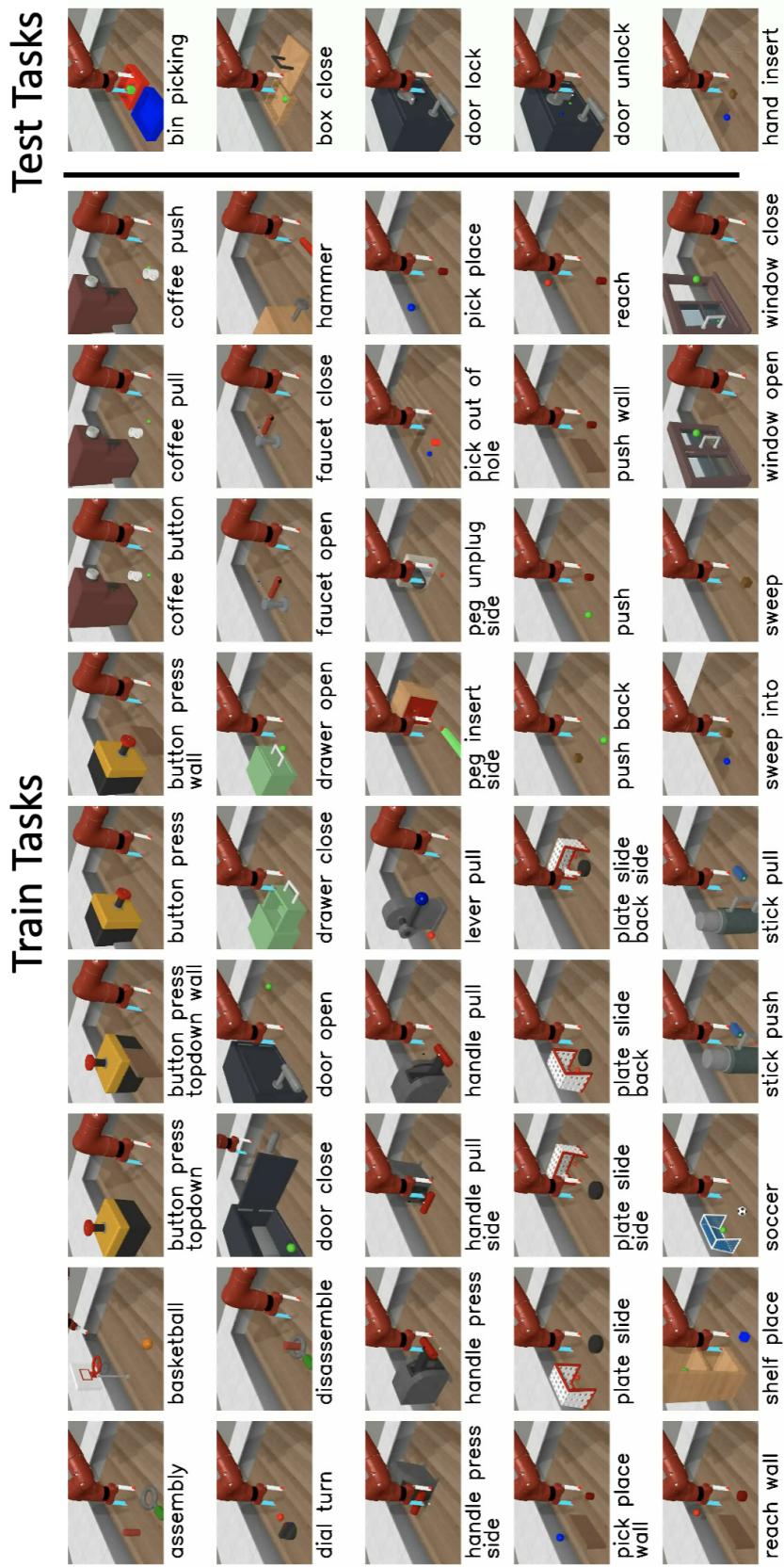


Figure 9: Enlarged image of Figure 1.

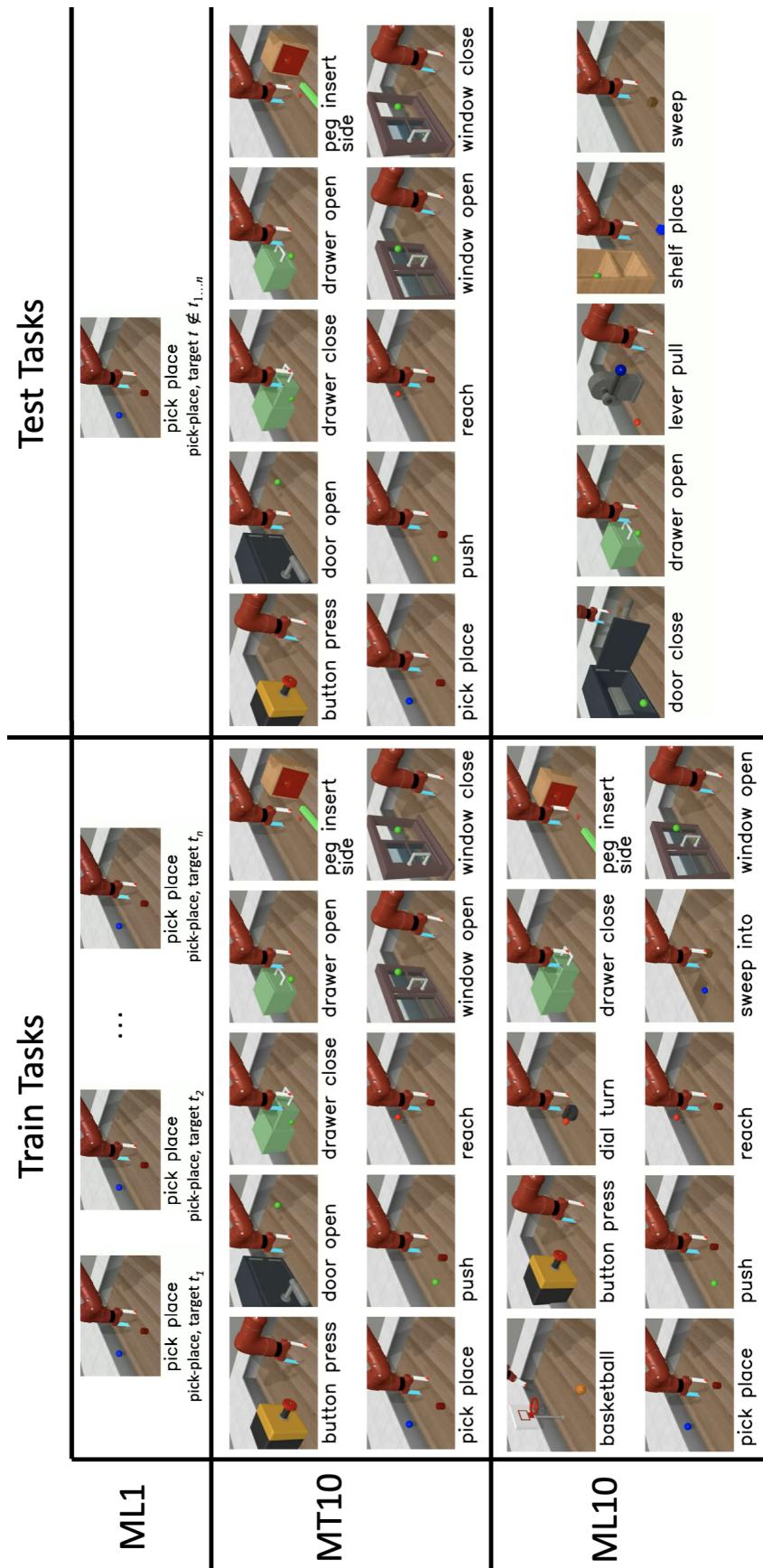


Figure 10: Enlarged image of Figure 3.

A Task Descriptions

In Table 2, we include a description of each of the 50 Meta-World tasks.

Task	Description
turn on faucet	Rotate the faucet counter-clockwise. Randomize faucet positions
sweep	Sweep a puck off the table. Randomize puck positions
assemble nut	Pick up a nut and place it onto a peg. Randomize nut and peg positions
turn off faucet	Rotate the faucet clockwise. Randomize faucet positions
push	Push the puck to a goal. Randomize puck and goal positions
pull lever	Pull a lever down 90 degrees. Randomize lever positions
turn dial	Rotate a dial 180 degrees. Randomize dial positions
push with stick	Grasp a stick and push a box using the stick. Randomize stick positions.
get coffee	Push a button on the coffee machine. Randomize the position of the coffee machine
pull handle side	Pull a handle up sideways. Randomize the handle positions
basketball	Dunk the basketball into the basket. Randomize basketball and basket positions
pull with stick	Grasp a stick and pull a box with the stick. Randomize stick positions
sweep into hole	Sweep a puck into a hole. Randomize puck positions
disassemble nut	pick a nut out of the a peg. Randomize the nut positions
place onto shelf	pick and place a puck onto a shelf. Randomize puck and shelf positions
push mug	Push a mug under a coffee machine. Randomize the mug and the machine positions
press handle side	Press a handle down sideways. Randomize the handle positions
hammer	Hammer a screw on the wall. Randomize the hammer and the screw positions
slide plate	Slide a plate into a cabinet. Randomize the plate and cabinet positions
slide plate side	Slide a plate into a cabinet sideways. Randomize the plate and cabinet positions
press button wall	Bypass a wall and press a button. Randomize the button positions
press handle	Press a handle down. Randomize the handle positions
pull handle	Pull a handle up. Randomize the handle positions
soccer	Kick a soccer into the goal. Randomize the soccer and goal positions
retrieve plate side	Get a plate from the cabinet sideways. Randomize plate and cabinet positions
retrieve plate	Get a plate from the cabinet. Randomize plate and cabinet positions
close drawer	Push and close a drawer. Randomize the drawer positions
press button top	Press a button from the top. Randomize button positions
reach	reach a goal position. Randomize the goal positions
press button top wall	Bypass a wall and press a button from the top. Randomize button positions
reach with wall	Bypass a wall and reach a goal. Randomize goal positions
insert peg side	Insert a peg sideways. Randomize peg and goal positions
pull	Pull a puck to a goal. Randomize puck and goal positions
push with wall	Bypass a wall and push a puck to a goal. Randomize puck and goal positions
pick out of hole	Pick up a puck from a hole. Randomize puck and goal positions
pick&place w/ wall	Pick a puck, bypass a wall and place the puck. Randomize puck and goal positions
press button	Press a button. Randomize button positions
pick&place	Pick and place a puck to a goal. Randomize puck and goal positions
pull mug	Pull a mug from a coffee machine. Randomize the mug and the machine positions
unplug peg	Unplug a peg sideways. Randomize peg positions
close window	Push and close a window. Randomize window positions
open window	Push and open a window. Randomize window positions
open door	Open a door with a revolving joint. Randomize door positions
close door	Close a door with a revolving joint. Randomize door positions
open drawer	Open a drawer. Randomize drawer positions
insert hand	Insert the gripper into a hole.
close box	Grasp the cover and close the box with it. Randomize the cover and box positions
lock door	Lock the door by rotating the lock clockwise. Randomize door positions
unlock door	Unlock the door by rotating the lock counter-clockwise. Randomize door positions
pick bin	Grasp the puck from one bin and place it into another bin. Randomize puck positions

Table 2: A list of all of the Meta-World tasks and a description of each task.

B Benchmark Verification with Single-Task Learning

In this section, we aim to verify that each of the benchmark tasks are individually solvable provided enough data. To do so, we consider two state-of-the-art single task reinforcement learning methods,

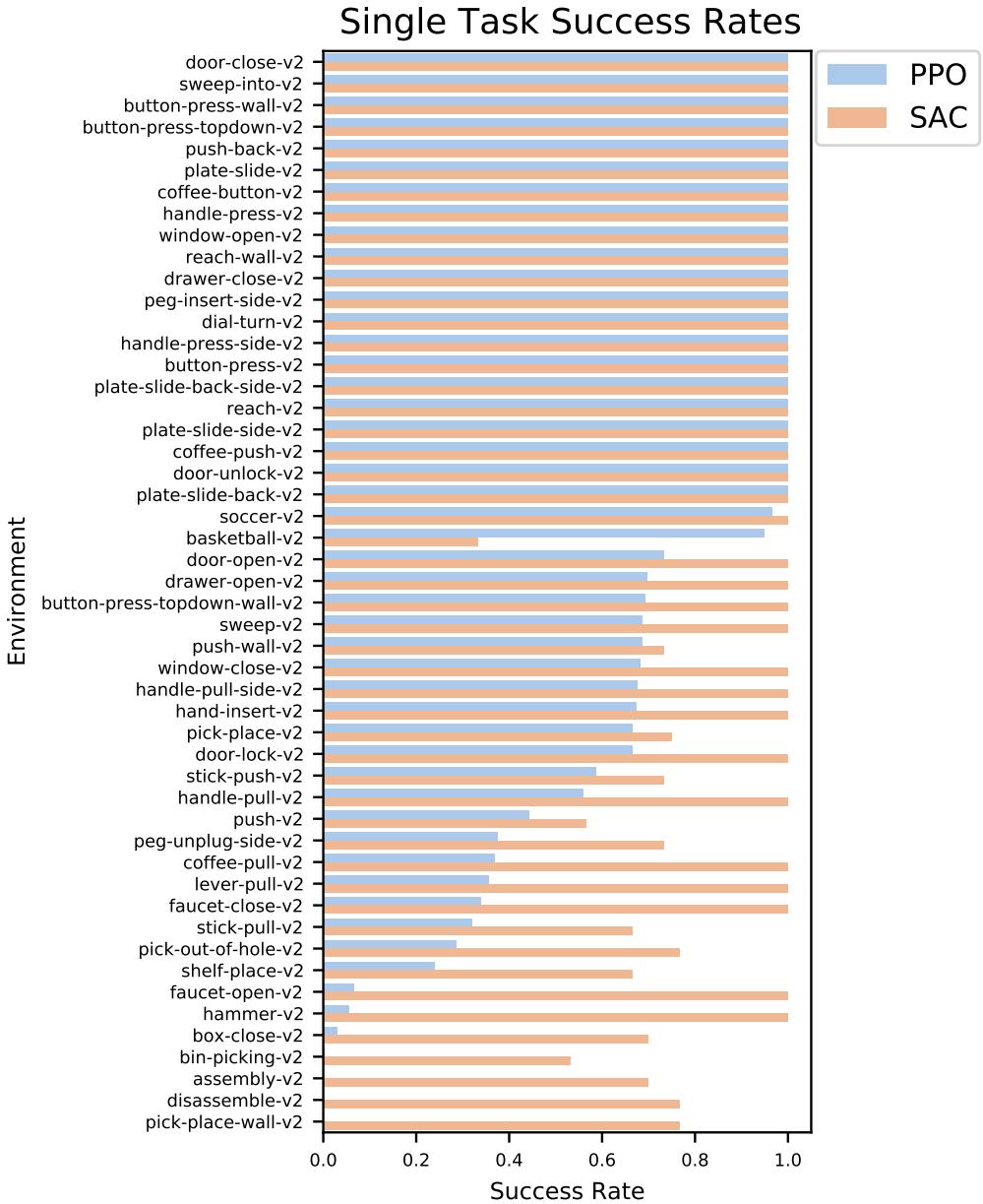


Figure 11: Performance of independent policies trained on individual tasks using soft actor-critic (SAC) and proximal policy optimization (PPO) on 3 seeds. We verify that SAC can solve all of the tasks and PPO can also solve most of the tasks.

proximal policy optimization (PPO) [55] and soft actor-critic (SAC) [57]. This evaluation is purely for validation of the tasks, and not an official evaluation protocol of the benchmark. Details of the hyperparameters are provided in Appendix D. The results of this experiment are illustrated in Figure 11. We indeed find that SAC can learn to perform all of the 50 tasks to some degree, while PPO can solve a large majority of the tasks.

C Learning curves

In evaluating meta-learning algorithms, we care not just about performance but also about efficiency, i.e. the amount of data required by the meta-training process. While the adaptation process for all algorithms is extremely efficient, requiring only a few trajectories, the meta-learning process can be very inefficient. In Figure 12, we show full learning curves of the three meta-learning methods on ML1. In Figure 15, we show full learning curves of MT10, ML10, MT50 and ML45. The MT10 and

MT50 learning curves show the efficiency of multi-task learning, a critical evaluation metric, since sample efficiency gains are a primary motivation for using multi-task learning. Unsurprisingly, we find that off-policy algorithms such as soft actor-critic are able to learn with substantially less data than on-policy algorithms.

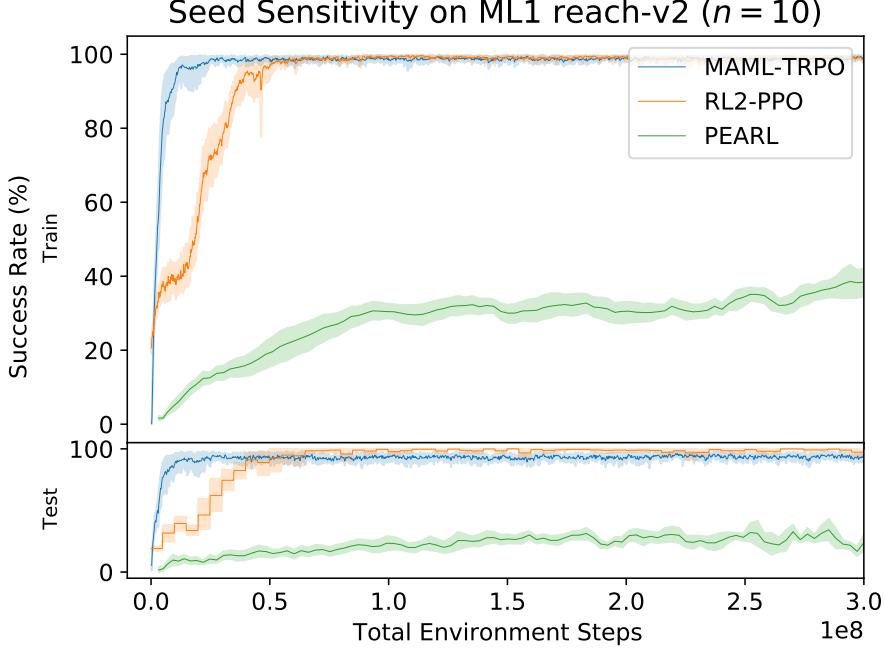


Figure 12: Comparison of PEARL, MAML, and RL^2 learning curves on ML-1 reach.

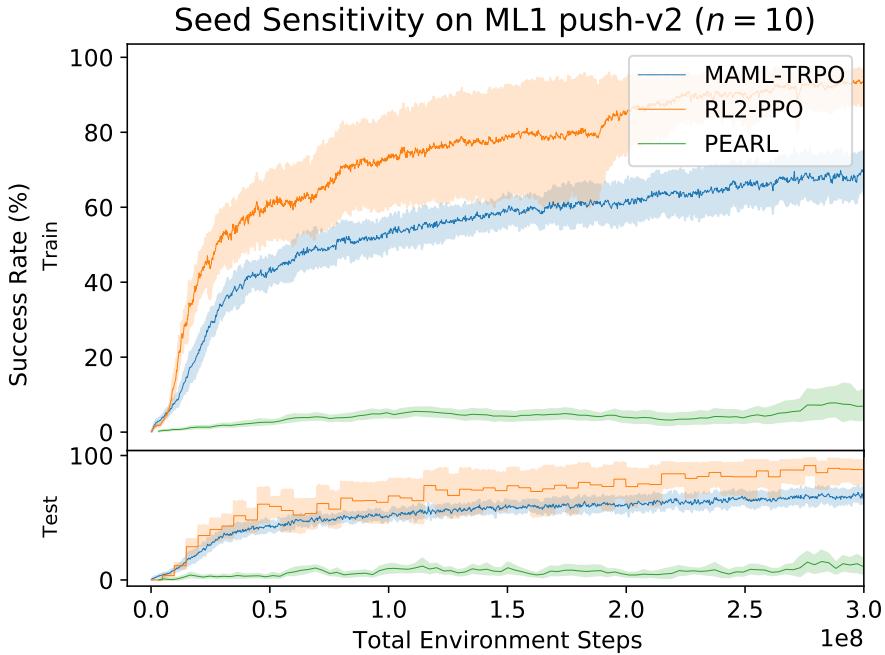


Figure 13: Comparison of PEARL, MAML, and RL^2 learning curves on ML-1 push.

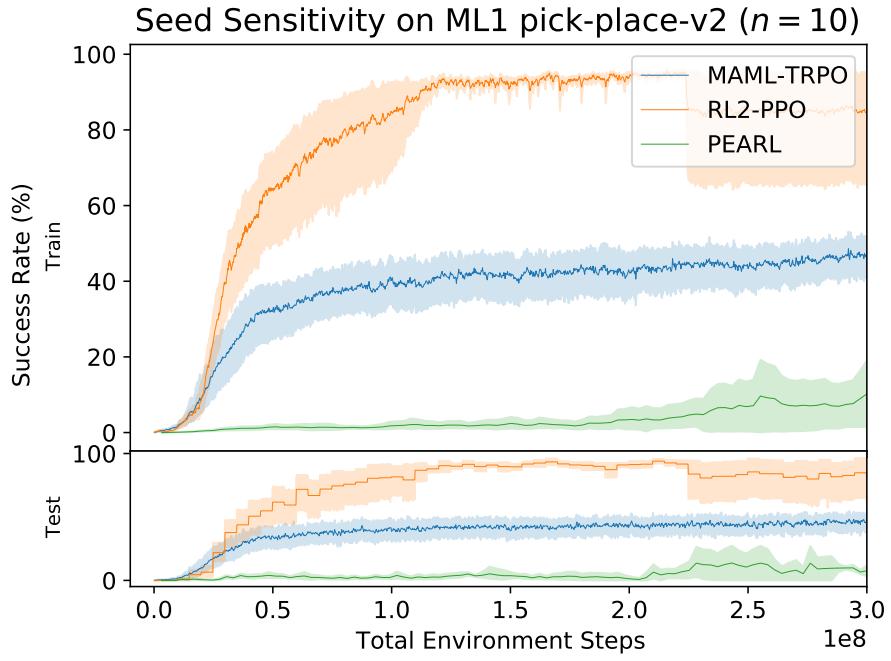


Figure 14: Comparison of PEARL, MAML, and RL^2 learning curves on the simplest evaluation, ML-1, where the methods need to adapt quickly to new object and goal positions within the one meta-training task.

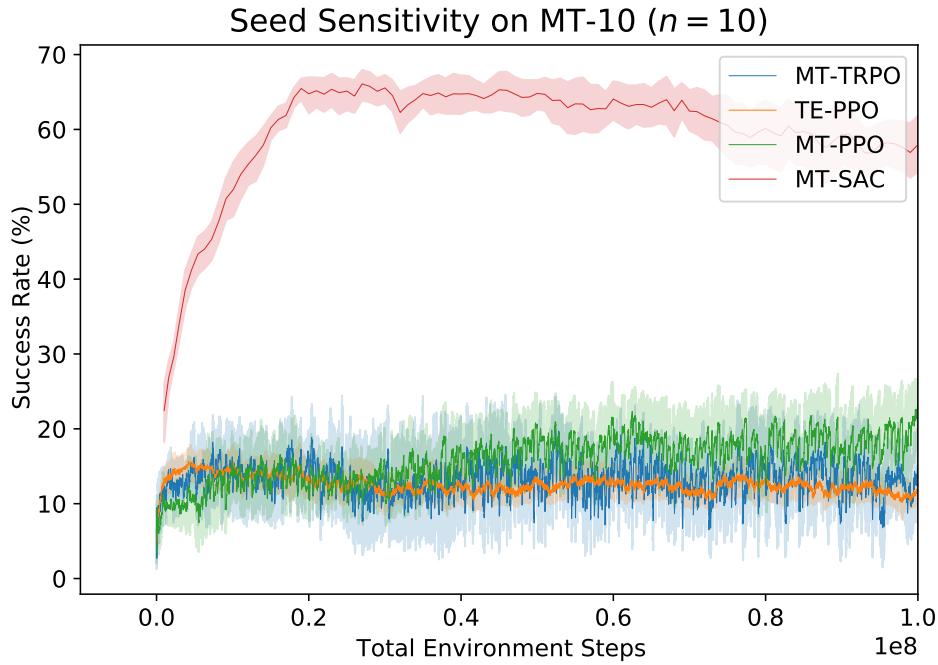


Figure 15: Comparison of MTRL algorithms on MT-10. MT-SAC vastly outperforms its on-policy counterparts in performance and sample efficiency.

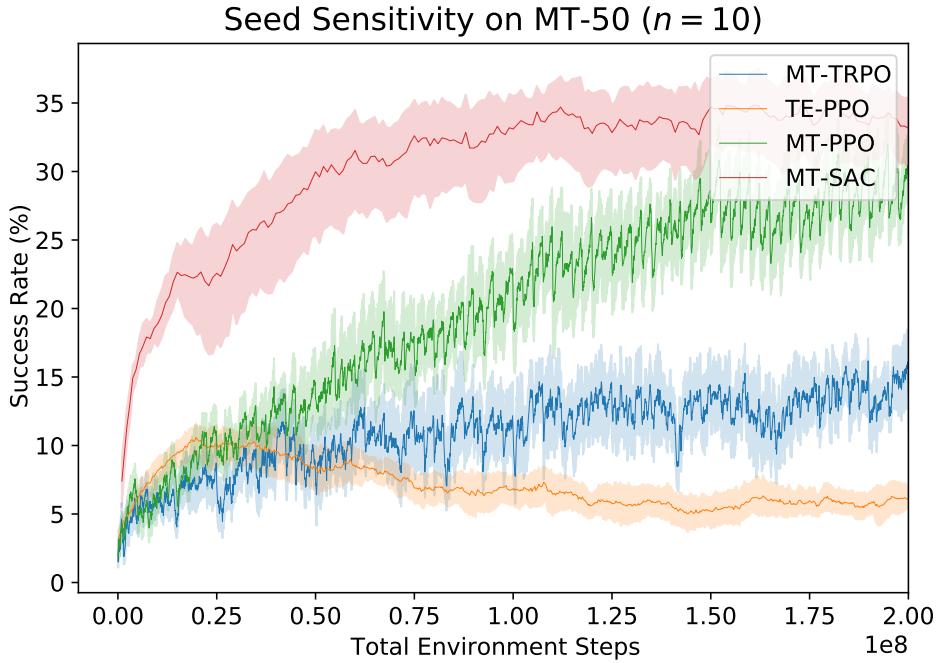


Figure 16: Comparison of MTRL algorithms on MT-50. MT-SAC vastly outperforms its on-policy counterparts in sample efficiency. Its performance tapers off, and with more training, MT-PPO outperforms it.

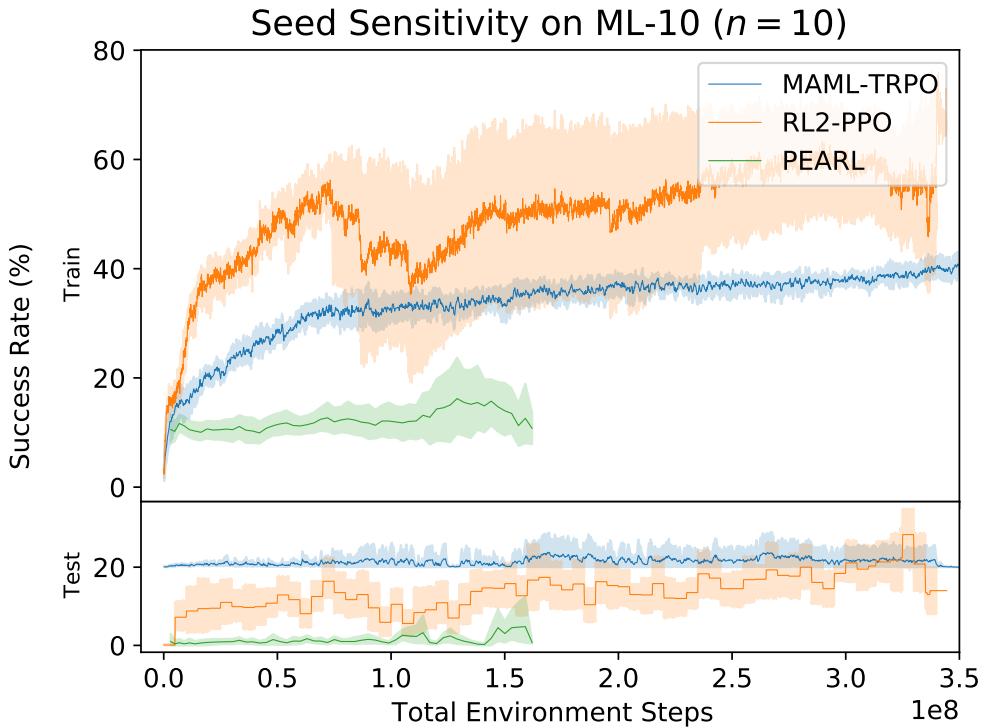


Figure 17: Performance of meta-RL algorithms on ML-10. RL^2 significantly outperforms other methods in terms of sample efficiency and performance on test tasks. MAML has better test performance early on, RL^2 outperforms it with more training.

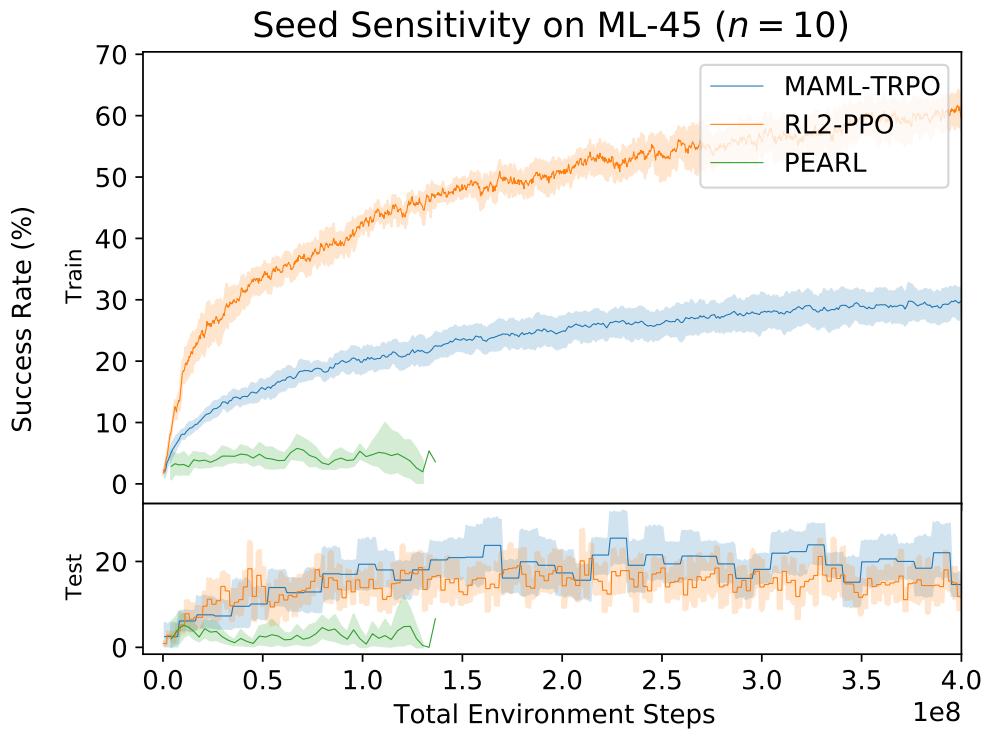


Figure 18: Learning curves of all methods on the ML-45 benchmark. Y-axis represents success rate averaged over tasks in percentage (%). The dashed lines represent asymptotic performances. PEARL underperforms MAML and RL². RL² significantly outperforms other methods in terms of sample efficiency and performance on train tasks. RL² and MAML have similar performance on test tasks.

D Hyperparameter Details

In this section, we provide hyperparameter values for each of the methods in our experimental evaluation.

D.1 Single Task SAC

Description	value	variable_name
Normal Hyperparameters		
Batch size	500	batch_size
Number of epochs	500	n_epochs
Path length per roll-out	500	max_path_length
Discount factor	0.99	discount
Algorithm-Specific Hyperparameters		
Policy hidden sizes	(256, 256)	hidden_sizes
Activation function of hidden layers	ReLU	hidden_nonlinearity
Policy learning rate	3×10^{-4}	policy_lr
Q-function learning rate	3×10^{-4}	qf_lr
Policy minimum standard deviation	e^{-20}	min_std
Policy maximum standard deviation	e^2	max_std
Gradient steps per epoch	500	gradient_steps_per_itr
Number of epoch cycles	40	epoch_cycles
Soft target interpolation parameter	5×10^{-3}	target_update_tau
Use automatic entropy Tuning	True	use_automatic_entropy_tuning

Table 3: Hyperparameters used for Garage experiments with Single Task SAC

D.2 Single Task PPO

Description	value	variable_name
Normal Hyperparameters		
Batch size	5,000	batch_size
Number of epochs	4,000	n_epochs
Path length per roll-out	500	max_path_length
Discount factor	0.99	discount
Algorithm-Specific Hyperparameters		
Policy mean hidden sizes	(128, 128)	hidden_sizes
Policy minimum standard deviation	0.5	min_std
Policy maximum standard deviation	1.5	max_std
Policy share standard deviation and mean network	True	std_share_network
Activation function of mean hidden layers	tanh	hidden_nonlinearity
Optimizer learning rate	5×10^{-4}	learning_rate
Likelihood ratio clip range	0.2	lr_clip_range
Advantage estimation λ	0.95	gae_lambda
Use layer normalization	False	layer_normalization
Entropy method	max	entropy_method
Loss function	surrogate_clip	pg_loss
Maximum number of epochs for update	256	max_epochs
Minibatch size for optimization	32	batch_size
Value Function Hyperparameters		
Policy hidden sizes	(128, 128)	hidden_sizes
Activation function of hidden layers	tanh	hidden_nonlinearity
Initial value for standard deviation	1	init_std
Use trust region constraint	False	use_trust_region
Normalize inputs	True	normalize_inputs
Normalize outputs	True	normalize_outputs

Table 4: Hyperparameters used for Garage experiments with Single Task PPO

Below we summarize in as much detail as possible the hyperparameters used for each experiment in this chapter. Seed values were individually chosen at random for each experiment.

D.3 MT-PPO

Description	MT10	MT50	variable_name
Normal Hyperparameters			
Batch size	100,000	500,000	batch_size
Number of epochs	10,000	10,000	n_epochs
Path length per roll-out	500	500	max_path_length
Discount factor	0.99	0.99	discount
Algorithm-Specific Hyperparameters			
Policy mean hidden sizes	(512, 512)	hidden_sizes	
Policy minimum standard deviation	0.5	0.5	min_std
Policy maximum standard deviation	1.5	1.5	max_std
Policy share standard deviation and mean network	True	True	std_share_network
Activation function of hidden layers	tanh	tanh	hidden_nonlinearity
Optimizer learning rate	5×10^{-4}	5×10^{-4}	learning_rate
Likelihood ratio clip range	0.2	0.2	lr_clip_range
Advantage estimation λ	0.97	0.97	gae_lambda
Use layer normalization	False	False	layer_normalization
Use trust region constraint	False	False	use_trust_region
Entropy method	max	max	entropy_method
Policy entropy coefficient	$5e - 3$	$5e - 3$	policy_ent_coeff
Loss function	surrogate_clip	surrogate_clip	pg_loss
Maximum number of epochs for update	16	16	max_epochs
Minibatch size for optimization	32	32	batch_size
Value Function Hyperparameters			
Value Function hidden sizes	(512, 512)	(512, 512)	hidden_sizes
Activation function of hidden layers	tanh	tanh	hidden_nonlinearity
Trainable standard deviation	True	True	learn_std
Initial value for standard deviation	1	1	init_std
Use layer normalization	False	False	layer_normalization
Use trust region constraint	False	False	use_trust_region
Normalize inputs	True	True	normalize_inputs
Normalize outputs	True	True	normalize_outputs

Table 5: Hyperparameters used for Garage experiments with Multi-Task PPO

D.4 MT-TRPO

Description	MT10	MT50	variable_name
Normal Hyperparameters			
Batch size	100,000	500,000	batch_size
Number of epochs	10,000	10,000	n_epochs
Path length per roll-out	500	500	max_path_length
Discount factor	0.99	0.99	discount
Algorithm-Specific Hyperparameters			
Policy mean hidden sizes	(512, 512)	hidden_sizes	
Policy minimum standard deviation	0.5	0.5	min_std
Policy maximum standard deviation	1.5	1.5	max_std
Policy share standard deviation and mean network	True	True	std_share_network
Activation function of hidden layers	tanh	tanh	hidden_nonlinearity
Advantage estimation λ	0.95	0.95	gae_lambda
Maximum KL divergence	1×10^{-2}	1×10^{-2}	max_kl_step
Number of CG iterations	10	10	cg_iters
Regularization coefficient	1×10^{-5}	1×10^{-5}	reg_coeff
Use layer normalization	False	False	layer_normalization
Use trust region constraint	False	False	use_trust_region
Entropy method	no_entropy	no_entropy	entropy_method
Loss function	surrogate	surrogate	pg_loss
Value Function Hyperparameters			
Hidden sizes	(512, 512)	(512, 512)	hidden_sizes
Activation function of hidden layers	tanh	tanh	hidden_nonlinearity
Trainable standard deviation	True	True	learn_std
Initial value for standard deviation	1	1	init_std
Use layer normalization	False	False	layer_normalization
Use trust region constraint	True	True	use_trust_region
Normalize inputs	True	True	normalize_inputs
Normalize outputs	True	True	normalize_outputs

Table 6: Hyperparameters used for Garage experiments with Multi-Task TRPO

D.5 MT-SAC

Description	MT10	MT50	variable_name
General Hyperparameters			
Batch size	5,000	25,000	batch_size
Number of epochs	500	500	epochs
Path length per roll-out	500	500	max_path_length
Discount factor	0.99	0.99	discount
Algorithm-Specific Hyperparameters			
Policy hidden sizes	(400, 400)	(400, 400)	hidden_sizes
Activation function of hidden layers	ReLU	ReLU	hidden_nonlinearity
Policy learning rate	3×10^{-4}	3×10^{-4}	policy_lr
Q-function learning rate	3×10^{-4}	3×10^{-4}	qf_lr
Policy minimum standard deviation	e^{-20}	e^{-20}	min_std
Policy maximum standard deviation	e^2	e^2	max_std
Gradient steps per epoch	500	500	gradient_steps_per_itr
Number of epoch cycles	200	40	epoch_cycles
Soft target interpolation parameter	5×10^{-3}	5×10^{-3}	target_update_tau
Use automatic entropy Tuning	True	True	use_automatic_entropy_tuning
Minimum Buffer Batch Size	1500	7500	min_buffer_size

Table 7: Hyperparameters used for Garage experiments with Multi-Task SAC

D.6 TE-PPO

Description	MT10	MT50	argument_name
General Hyperparameters			
Batch size	50,000	250,000	batch_size
Number of epochs	4,000	2,000	n_epochs
Algorithm-Specific Hyperparameters			
Policy hidden sizes	(32, 16)	(32, 16)	hidden_sizes
Activation function of hidden layers	tanh	tanh	hidden_nonlinearity
Likelihood ratio clip range	0.2	0.2	lr_clip_range
Latent dimension	4	4	latent_length
Inference window length	6	6	inference_window
Embedding maximum standard deviation	0.2	0.2	embedding_max_std
Policy entropy coefficient	$2e - 2$	$2e - 2$	policy_ent_coeff
Value function	Gaussian MLP fit with observations, latent variables and returns		baseline

Table 8: Hyperparameters used for Garage experiments with Task Embeddings PPO

D.7 MAML

Description	ML1	ML10	ML45	argument_name
Meta-/Multi-Task Hyperparameters				
Meta-batch size	20	20	45	meta_batch_size
Roll-outs per task	10	10	20	rollouts_per_task
General Hyperparameters				
Path length per roll-out	500	500	500	max_path_length
Discount factor	0.99	0.99	0.99	discount
Algorithm-specific Hyperparameters				
Policy hidden sizes	(128, 128)	(128, 128)	(128, 128)	hidden_sizes
Activation function of hidden layers	tanh	tanh	tanh	hidden_nonlinearity
Activation function of output layer	tanh	tanh	tanh	output_nonlinearity
Inner algorithm learning rate	1×10^{-4}	1×10^{-4}	1×10^{-4}	inner_lr
Optimizer learning rate	1×10^{-3}	1×10^{-3}	1×10^{-3}	outer_lr
Maximum KL divergence	1×10^{-2}	1×10^{-2}	1×10^{-2}	max_kl_step
Number of inner gradient updates	1	1	1	num_grad_update
Policy entropy coefficient	5×10^{-5}	5×10^{-5}	5×10^{-5}	policy_ent_coeff

Table 9: Hyperparameters used for Garage experiments with MAML

D.8 RL²

Description	ML1	ML10	ML45	argument_name
Meta-/Multi-Task Hyperparameters				
Meta-batch size	25	10	25	meta_batch_size
Roll-outs per task	10	10	10	rollouts_per_task
General Hyperparameters				
Path length per roll-out	500	500	500	max_path_length
Discount factor	0.99	0.99	0.99	discount
Algorithm-Specific Hyperparameters				
Policy hidden sizes	(256,)	(256,)	(256,)	hidden_sizes
Activation function of hidden layers	tanh	tanh	tanh	hidden_nonlinearity
Activation function of recurrent layers	sigmoid	sigmoid	sigmoid	recurrent_nonlinearity
Optimizer learning rate	5×10^{-4}	5×10^{-4}	5×10^{-4}	optimizer_lr
Likelihood ratio clip range	0.2	0.2	0.2	lr_clip_range
Advantage estimation λ	0.95	0.95	0.95	gae_lambda
Optimizer maximum epochs	10	10	10	optimizer_max_epochs
RNN cell type used in Policy	GRU	GRU	GRU	cell_type
Value function	Linear feature baseline			baseline
Policy entropy coefficient	5×10^{-6}	5×10^{-6}	5×10^{-6}	policy_ent_coeff
Minimum policy standard deviation	0.5	0.5	0.5	min_std
Maximum policy standard deviation	0.5	0.5	0.5	max_std

Table 10: Hyperparameters used for Garage experiments with RL²

D.9 PEARL

Description	ML1	ML10	ML45	argument_name
Meta-/Multi-Task Hyperparameters				
Meta-batch size	16	10	45	meta_batch_size
Tasks sampled per epoch	15	10	45	num_tasks_sample
Number of independent evaluations		5		num_evals
Steps sampled per evaluation	450	1,650	1,650	num_steps_per_eval
General Hyperparameters				
Batch size	500	1,000	1,000	batch_size
Path length per roll-out		500		max_path_length
Reward scale		10,000		reward_scale
Discount factor		0.99		discount
Algorithm-Specific Hyperparameters				
Policy hidden sizes	(300, 300, 300)			net_size
Activation function of hidden layers	ReLU			hidden_nonlinearity
Policy learning rate	3×10^{-4}			policy_lr
Q-function learning rate	3×10^{-4}			qf_lr
Value function learning rate	3×10^{-4}			vf_lr
Context learning rate	3×10^{-4}			context_lr
Latent dimension	7			latent_dimension
Policy mean regularization coefficient	1×10^{-3}			policy_mean_reg_coeff
Policy standard deviation regularization coefficient	1×10^{-3}			policy_std_reg_coeff
Soft target interpolation parameter	5×10^{-3}			soft_target_tau
KL λ	0.01			KL_lambda
Use information bottleneck	True			use_information_bottleneck
Use next observation in context	False			use_next_observation_in_context
Gradient steps per epoch	1	60	14	num_steps_per_epoch
Steps sampled in the initial epoch	1,000	5,000	22,500	num_initial_steps
Prior steps sampled per epoch	2500			num_steps_prior
Posterior steps sampled per epoch	2500			num_steps_posterior
Extra posterior steps sampled per epoch	2500			num_extra_steps_posterior
Embedding batch size	250			embedding_batch_size
Embedding minibatch size	250			embedding_mini_batch_size

Table 11: Hyperparameters used for Garage experiments with PEARL

E Reward Functions and Single-Task Results

E.1 Reward Functions

The variables that will be discussed are the following:

$$O \in \mathbb{R}^3 : \text{object position}$$

$$h \in \mathbb{R}^3 : \text{hand/gripper position}$$

$$t \in \mathbb{R}^3 : \text{target/goal position}$$

$$h_l \in \mathbb{R}^3 : \text{position of the left hand/gripper pad}$$

$$h_r \in \mathbb{R}^3 : \text{position of the right hand/gripper pad}$$

$$O_i \in \mathbb{R}^3 : \text{initial position of the object}$$

$$h_i \in \mathbb{R}^3 : \text{initial position of the hand/gripper}$$

$$g \in \mathbb{R} : \text{gripper closed/open amount}$$

The following tolerance function is used frequently:

$$L(x, b_{min}, b_{max}, m) = \begin{cases} 1 & b_{min} \leq x \leq b_{max} \\ S\left(\frac{b_{min}-x}{m}, 0.1\right) & x < b_{min} \\ S\left(\frac{x-b_{max}}{m}, 0.1\right) & x \geq b_{max} \end{cases}$$

Where S is defined to be a long-tail sigmoid:

$$S(a_1, a_2) = \left(\left(\frac{1}{a_2 - 1} - 1 \right) a_1^2 + 1 \right)^{-1}$$

With these basics in place, we define a caging tensor that describes behaviour in an axis which intersects the gripper's actuated fingers (in code, the Y axis):

$$C_{LR}(c_1, c_2) = L\left(\left|\begin{bmatrix} h_{L,(y)} \\ h_{R,(y)} \end{bmatrix} - o_{(y)}\right|, c_1, c_2, \left|\left|\begin{bmatrix} h_{L,(y)} \\ h_{R,(y)} \end{bmatrix} - o_{i,(y)}\right| - c_2\right|\right)$$

A similar caging value describes behaviour in the other two axes (in code, X and Z axes):

$$C_P(c_3) = L(\|o_{(xz)} - h_{(xz)}\|, 0, c_3, \|o_{i,(xz)} - h_{i,(xz)}\| - c_3)$$

These get lumped together as follows (T_{H_0} is the Hamacher product):

$$C(c_1, c_2, c_3) = T_{H_0}(T_{H_0}(C_{LR,(0)}, C_{LR,(1)}), C_P(c_3))$$

The caging reward has two modes: medium density and high density. The arguments c_1, c_2, c_3 are passed to C

$$R_{cage,dense}(c_1, c_2, c_3) = \begin{cases} 0.5(C + T_{H_0}(C, g)) & C > 0.97 \\ 0.5C & \text{otherwise} \end{cases}$$

$$R_{cage}(c_1, c_2, c_3, c_4) = \begin{cases} 0.5(L(\|o - h\|, 0, c_4, \|o - h_i\|) + T_{H_0}(C, g)) & C > 0.97 \\ 0.5L(\|o - h\|, 0, c_4, \|o - h_i\|) & \text{otherwise} \end{cases}$$

In each set of expressions given below, the arguments passed to R_{cage} or $R_{cage,dense}$ correspond to $[c_1, c_2, c_3, \dots]$. The caging reward also considers $[h, h_i, o, o_i]$ as described on the previous page, but these arguments are omitted for brevity.

If computation involves a parameter A , understand that A is non-zero *iff* the Sawyer successfully grasps the object. As such, A serves as a post-grasp guidance term.

Common patterns include $A + T_{H_0}(R_{cage}, L(t - o, \dots))$, $T_{H_0}(1 - g, L(o - h, \dots))$, and $L(t - o, \dots) + L(o - h, \dots)$. As a general rule, rewards for simple tasks consist of summed tolerances, while more difficult tasks add complexity in the form of Hamacher Products. The Hamacher Products combine tolerances, grip effort, and/or R_{cage} to produce a smooth, dense reward.

E.1.1 Basketball

$$A = \mathbb{I}_{\|o-h\| < 0.035 \& g > 0 \& o_{(z)} - o_{i(z)} > 0.01} \cdot (1 + L(\|\langle 1, 1, 2 \rangle \cdot (t - o)\|, 0, 0.08, \|\langle 1, 1, 2 \rangle \cdot (t - o_i)\|))$$

$$R = \begin{cases} A + T_{H_0}(R_{cage,dense}(0.025, 0.06, 0.005), \\ \quad L(\|\langle 1, 1, 2 \rangle \cdot (t - o)\|, 0, 0.08, \|\langle 1, 1, 2 \rangle \cdot (t - o_i)\|)) & \|\langle 1, 1, 2 \rangle \cdot (t - o)\| \geq 0.08 \\ 10 & otherwise \end{cases}$$

E.1.2 Button Press Top Down

$$R = \begin{cases} 5T_{H_0}(1 - g, L(\|o - h\|, 0, 0.01, \|o - h_i\|)) & \|o - h\| > 0.03 \\ 5T_{H_0}(1 - g, L(\|o - h\|, 0, 0.01, \|o - h_i\|)) + 5L(|t_{(z)} - o_{(z)}|, 0, 0.005, |t_{(z)} - o_{i,(z)}|) & otherwise \end{cases}$$

E.1.3 Button Press Top Down Wall

$$R = \begin{cases} 5T_{H_0}(1 - g, L(\|o - h\|, 0, 0.01, \|o - h_i\|)) & \|o - h\| > 0.03 \\ 5T_{H_0}(1 - g, L(\|o - h\|, 0, 0.01, \|o - h_i\|)) + 5L(|t_{(z)} - o_{(z)}|, 0, 0.005, |t_{(z)} - o_{i,(z)}|) & otherwise \end{cases}$$

E.1.4 Button Press

$$R = \begin{cases} 2T_{H_0}(g, L(\|o - h\|, 0, 0.05, \|o - h_i\|)) & \|o - h\| > 0.05 \\ 2T_{H_0}(g, L(\|o - h\|, 0, 0.05, \|o - h_i\|)) + 8L(|t_{(y)} - o_{(y)}|, 0, 0.005, |t_{(y)} - o_{i,(y)}|) & otherwise \end{cases}$$

E.1.5 Button Press Wall

$$R = \begin{cases} 2T_{H_0}(1 - g, L(\|o - h\|, 0, 0.01, \|o - h_i\|)) & \|o - h\| > 0.07 \\ 4 + 2g + 4(L(|t_{(y)} - o_{(y)}|, 0, 0.005, |t_{(y)} - o_{i,(y)}|)^2) & otherwise \end{cases}$$

E.1.6 Coffee Button

$$R = \begin{cases} 2T_{H_0}(g, L(\|o - h\|, 0, 0.05, \|o - h_i\|)) & \|o - h\| > 0.05 \\ 2T_{H_0}(g, L(\|o - h\|, 0, 0.05, \|o - h_i\|)) + 8L(|t_{(y)} - o_{(y)}|, 0, 0.005, |t_{(y)} - o_{i,(y)}|) & otherwise \end{cases}$$

E.1.7 Coffee Pull

$$A = \mathbb{I}_{\|o-h\| < 0.04 \& g > 0} \cdot (1 + 5L(\|\langle 2, 2, 1 \rangle \cdot (t - o)\|, 0, 0.05, \|\langle 2, 2, 1 \rangle \cdot (t - o_i)\|))$$

$$R = \begin{cases} A + T_{H_0}(R_{cage}(0.02, 0.05, 0.05, 0.04), \\ \quad L(\|\langle 2, 2, 1 \rangle \cdot (t - o)\|, 0, 0.05, \|\langle 2, 2, 1 \rangle \cdot (t - o_i)\|)) & \|\langle 2, 2, 1 \rangle \cdot (t - o)\| \geq 0.05 \\ 10 & otherwise \end{cases}$$

E.1.8 Coffee Push

$$A = \mathbb{I}_{\|o-h\| < 0.04 \& g > 0} \cdot (1 + 5L(\|\langle 2, 2, 1 \rangle \cdot (t - o)\|, 0, 0.05, \|\langle 2, 2, 1 \rangle \cdot (t - o_i)\|))$$

$$R = \begin{cases} A + T_{H_0}(R_{cage}(0.02, 0.05, 0.05, 0.04), \\ \quad L(\|\langle 2, 2, 1 \rangle \cdot (t - o)\|, 0, 0.05, \|\langle 2, 2, 1 \rangle \cdot (t - o_i)\|)) & \|\langle 2, 2, 1 \rangle \cdot (t - o)\| \geq 0.05 \\ 10 & otherwise \end{cases}$$

E.1.9 Door Close

$$R = \begin{cases} 6L(\|t - o\|, 0, 0.05, \|t - o_i\|) + 3L(\|t - h\|, 0, 0.012, 0.1 + \|h_i - o\|) & \|t - o\| \geq 0.05 \\ 10 & otherwise \end{cases}$$

E.1.10 Door Lock

$$R = 2T_{H_0}(g, L(\|\langle 1, 4, 2 \rangle \cdot (o - h)\|, 0, 0.01, \|\langle 1, 4, 2 \rangle \cdot (o - h_i)\|)) + 8L(|t_{(z)} - o_{i,(z)}|, 0, 0.005, 0.1)$$

E.1.11 Door Unlock

$$R = \begin{cases} 2L(\|\langle 1, 4, 2 \rangle \cdot (o - h + \langle 0, 0.055, 0.07 \rangle)\|), \\ 0, \\ 0.02, \\ \|\langle 1, 4, 2 \rangle \cdot (o_i - h_i + \langle 0, 0.055, 0.07 \rangle)\|) + 8L(|t_{(x)} - o_{i,(x)}|, 0, 0.005, 0.1) \end{cases}$$

E.1.12 Door Open

$$\begin{aligned} alt &= \mathbb{I}_{\|h_{(xy)} - o_{(xy)}\| > 0.12} \cdot (0.4 + 0.04 \log (\|h_{(xy)} - o_{(xy)}\| - 0.12)) \\ ready &= \begin{cases} T_{H_0}(L(\|h - o - \langle 0.05, 0.03, -0.01 \rangle\|, 0, 0.06, 0.5), L(alt - h_{(z)}, 0, 0.01, \frac{alt}{2}),) & h_{(z)} < alt \\ L(\|h - o - \langle 0.05, 0.03, -0.01 \rangle\|, 0, 0.06, 0.5) & otherwise \end{cases} \\ R &= \begin{cases} 2T_{H_0}(g, ready) + 8(0.2\mathbb{I}_{o_{(\theta)} < 0.03} + 0.8L(o_{(\theta)} + \frac{2\pi}{3}, 0, 0.5, \frac{\pi}{3})) & |t_{(x)} - o_{(x)}| > 0.08 \\ 10 & otherwise \end{cases} \end{aligned}$$

E.1.13 Box Close

$$\begin{aligned} alt &= \mathbb{I}_{\|h_{(xy)} - o_{(xy)}\| > 0.02} \cdot (0.4 + 0.04 \log (\|h_{(xy)} - o_{(xy)}\| - 0.02)) \\ ready &= \begin{cases} T_{H_0}(L(\|h - o\|, 0, 0.02, 0.5), L(alt - h_{(z)}, 0, 0.01, \frac{alt}{2}),) & h_{(z)} < alt \\ L(\|h - o\|, 0, 0.02, 0.5) & otherwise \end{cases} \\ R &= \begin{cases} 2T_{H_0}(\frac{g+1}{2}, ready) + 8(0.2\mathbb{I}_{o_{(z)} > 0.04} + 0.8L(\langle 1, 1, 3 \rangle \|t - o\|, 0, 0.05, 0.25)) & |t - o| \geq 0.08 \\ 10 & otherwise \end{cases} \end{aligned}$$

E.1.14 Drawer Open

$$R = 5(L(\|t - o\|, 0, 0.02, 0.2) + L(\|(o - h) \cdot \langle 3, 3, 1 \rangle\|, 0, 0.01, \|(o_i - h_i) \cdot \langle 3, 3, 1 \rangle\|))$$

E.1.15 Drawer Close

$$R = \begin{cases} T_{H_0}(L(\|t - o\|, 0, 0.05, \|t - o_i\| - 0.05), T_{H_0}(g, L(\|o - h\|, 0, 0.005, \|o_i - h_i\| - 0.005))) & \|t - o\| > 0.065 \\ 10 & otherwise \end{cases}$$

E.1.16 Faucet Close

$$R = \begin{cases} 4L(\|o - h\|, 0, 0.01, \|o_i - h_i\| - 0.01) + 6L(\|t - o\|, 0, 0.07, \|t - o_i\| - 0.07) & \|t - o\| > 0.07 \\ 10 & otherwise \end{cases}$$

E.1.17 Faucet Open

$$R = \begin{cases} (4L(\|o - h + \langle -0.04, 0, .03 \rangle\|, 0, 0.01, \|o_i - h_i\| - 0.01) \\ + 6L(\|t - o + \langle -0.04, 0, .03 \rangle\|, 0, 0.07, \|t - o_i\| - 0.07)) & \|t - o + \langle -0.04, 0, .03 \rangle\| > 0.07 \\ 10 & otherwise \end{cases}$$

E.1.18 Hand Insert

$$A = \mathbb{I}_{\|o - h\| < 0.02 \& g > 0} \cdot (1 + 7L(\|t - o\|, 0, 0.05, \|t - o_i\|))$$

$$R = \begin{cases} A + T_{H_0}(R_{cage,dense}(0.015, 0.05, 0.005), L(\|t - o\|, 0, 0.05, \|t - o_i\|)) & \|t - o\| > 0.05 \\ 10 & otherwise \end{cases}$$

E.1.19 Pick Place

$$A = \mathbb{I}_{\|o - h\| < 0.02 \& g > 0 \& o_{(z)} > 0.01} \cdot (1 + 5L(\|t - o\|, 0, 0.05, \|t - o_i\|))$$

$$R = \begin{cases} A + T_{H_0}(R_{cage,dense}(0.015, 0.05, 0.005), L(\|t - o\|, 0, 0.05, \|t - o_i\|)) & \|t - o\| > 0.05 \\ 10 & otherwise \end{cases}$$

E.1.20 Pick Out Of Hole

A funnel-shaped surface guides the gripper as it seeks to grab and lift the object; this prevents the gripper from running into the side of the hole in the table. The height (or "altitude") of this surface is given by alt since the variables h and z are already used.

$$alt = \mathbb{I}_{\|h_{(xy)} - o_{i,(xy)}\| > 0.03} \cdot (0.15 + 0.015 \log (\|h_{(xy)} - o_{i,(xy)}\| - 0.03))$$

$$A = \mathbb{I}_{\|o-h\| < 0.04 \text{ & } g < 0.33 \text{ & } o_{(z)} - o_{i,(z)} > 0.02}$$

$$\bullet \left(1 + 5T_{H_0} \left(\frac{L(\|t - o\|, 0, 0.02, \|t - o_i\|),}{\max(\mathbb{I}_{h_{(z)} > alt}, L(alt - h_{(z)}, 0, 0.01, 0.02))} \right) \right)$$

$$R = \begin{cases} A + T_{H_0} (R_{cage,dense}(0.015, 0.05, 0.005), L(\|t - o\|, 0, 0.05, \|t - o_i\|)) & \|t - o\| > 0.05 \\ 10 & otherwise \end{cases}$$

E.1.21 Plate Slide Back Side

$$A = \mathbb{I}_{\|o-h\| < 0.07 \text{ & } h_{(z)} \leq 0.03}$$

$$R = \begin{cases} A \cdot (2 + 7L(\|t - o\|, 0, 0.05, \|t - o_i\|)) + (1 - A) \cdot 1.5L(\|o - h\|, 0, 0.05, \|o_i - h_i\| - 0.05) & \|t - o\| > 0.05 \\ 10 & otherwise \end{cases}$$

E.1.22 Plate Slide Back

$$A = \mathbb{I}_{\|o-h\| < 0.07 \text{ & } h_{(z)} \leq 0.03}$$

$$R = \begin{cases} A \cdot (2 + 7L(\|t - o\|, 0, 0.05, \|t - o_i\|)) + (1 - A) \cdot 1.5L(\|o - h\|, 0, 0.05, \|o_i - h_i\| - 0.05) & \|t - o\| > 0.05 \\ 10 & otherwise \end{cases}$$

E.1.23 Plate Slide Side

$$A = \mathbb{I}_{\|o-h\| < 0.07 \text{ & } h_{(z)} \leq 0.03}$$

$$R = \begin{cases} A \cdot (2 + 7L(\|t - o\|, 0, 0.05, \|t - o_i\|)) + (1 - A) \cdot 1.5L(\|o - h\|, 0, 0.05, \|o_i - h_i\| - 0.05) & \|t - o\| > 0.05 \\ 10 & otherwise \end{cases}$$

E.1.24 Plate Slide

$$R = \begin{cases} 8T_{H_0} (L(\|t - o\|, 0, 0.05, \|t - o_i\|), L(\|o - h\|, 0, 0.05, \|o_i - h_i\|)) & \|t - o\| \geq 0.05 \\ 10 & otherwise \end{cases}$$

E.1.25 Handle Press Side

$$R = \begin{cases} 10T_{H_0} (L(|t_{(z)} - o_{(z)}|, 0, 0.05, |t_{(z)} - o_{i,(z)}|), L(\|o - h\|, 0, 0.02, \|o_i - h_i\| - 0.02)) & \|t - o\| > 0.05 \\ 10 & otherwise \end{cases}$$

E.1.26 Handle Press

$$R = \begin{cases} 10T_{H_0} (L(|t_{(z)} - o_{(z)}|, 0, 0.05, |t_{(z)} - o_{i,(z)}|), L(\|o - h\|, 0, 0.02, \|o_i - h_i\| - 0.02)) & \|t - o\| > 0.05 \\ 10 & otherwise \end{cases}$$

E.1.27 Handle Pull

$$A = \mathbb{I}_{\|o-h\| < 0.035 \text{ & } g > 0 \text{ & } o_{(z)} - o_{i,(z)} > 0.01} \cdot (1 + 5L(|t_{(z)} - o_{(z)}|, 0, 0.05, |t_{(z)} - o_{i,(z)}|))$$

$$R = \begin{cases} A + T_{H_0} (R_{cage,dense}(0.022, 0.05, 0.01), L(|t_{(z)} - o_{(z)}|, 0, 0.05, |t_{(z)} - o_{i,(z)}|)) & \|t - o\| > 0.05 \\ 10 & otherwise \end{cases}$$

E.1.28 Handle Pull Side

$$A = \mathbb{I}_{\|o-h\| < 0.035 \text{ & } g > 0 \text{ & } o_{(z)} - o_{i,(z)} > 0.01} \cdot (1 + 5L(|t_{(z)} - o_{(z)}|, 0, 0.05, |t_{(z)} - o_{i,(z)}|))$$

$$R = \begin{cases} A + T_{H_0} (R_{cage,dense}(0.032, 0.06, 0.01), L(|t_{(z)} - o_{(z)}|, 0, 0.05, |t_{(z)} - o_{i,(z)}|)) & \|t - o\| > 0.05 \\ 10 & otherwise \end{cases}$$

E.1.29 Reach

$$R = 10L(\|t - h\|, 0, 0.05, \|t - h_i\|)$$

E.1.30 Reach Wall

$$R = 10L(\|t - h\|, 0, 0.05, \|t - h_i\|)$$

E.1.31 Push

$$A = \mathbb{I}_{\|o-h\| < 0.02 \text{ & } g > 0}$$

$$R = \begin{cases} (A + 1) \cdot R_{cage,dense}(0.015, 0.05, 0.005) + A \cdot (1 + 5L(\|t - o\|, 0, 0.05, \|t - o_i\|)) & \|t - o\| > 0.05 \\ 10 & otherwise \end{cases}$$

E.1.32 Sweep Into Goal

Note: This technically uses a $R_{cage,dense}$ function with slightly different margin parameters than the one described above (they are constant rather than dynamic), but the behaviour is mostly the same.

$$R = \begin{cases} (2R_{cage,dense}(0.02, 0.05, 0.01) \\ + 2T_{H_0}(R_{cage,dense}(0.02, 0.05, 0.01), L(\|t - o\|, 0, 0.05, \|t - o_i\|))) & \|t - o\| > 0.05 \\ 10 & otherwise \end{cases}$$

E.1.33 Sweep

Note: This technically uses a $R_{cage,dense}$ function with slightly different margin parameters than the one described above (they are constant rather than dynamic), but the behaviour is mostly the same.

$$R = \begin{cases} (2R_{cage,dense}(0.02, 0.05, 0.01) \\ + 2T_{H_0}(R_{cage,dense}(0.02, 0.05, 0.01), L(\|t - o\|, 0, 0.05, \|t - o_i\|))) & \|t - o\| > 0.05 \\ 10 & otherwise \end{cases}$$

E.1.34 Push Back

Note: This technically uses a $R_{cage,dense}$ function with slightly different margin parameters than the one described above (they are constant rather than dynamic), but the behaviour is mostly the same.

$$A = \mathbb{I}_{\|o-h\| < 0.01 \text{ & } 0 < g < 0.55 \text{ & } \|t - o_i\| - \|t - o\| > 0.01} \cdot (1 + 5L(\|t - o\|, 0, 0.05, \|t - o_i\|))$$

$$R = \begin{cases} A + T_{H_0}(R_{cage,dense}(0.01, 0.05, 0.01), L(\|t - o\|, 0, 0.05, \|t - o_i\|)) & \|t - o\| > 0.05 \\ 10 & otherwise \end{cases}$$

E.1.35 Window Open

$$R = 10T_{H_0}(L(|t_{(x)} - o_{(x)}|, 0, 0.05, |t_{(x)} - o_{i,(x)}|), L(\|o - h\|, 0, 0.02, \|o_i - h_i\| - 0.02))$$

E.1.36 Window Close

$$R = 10T_{H_0}(L(|t_{(x)} - o_{(x)}|, 0, 0.05, |t_{(x)} - o_{i,(x)}|), L(\|o - h\|, 0, 0.02, \|o_i - h_i\| - 0.02))$$

E.1.37 Dial Turn

$$\begin{aligned} R = 10T_{H_0}(L(\|t - o\|, 0, 0.05, \|t - o_i\| - 0.05), \\ T_{H_0}(g, \\ L(\|o - h + \langle 0.05, 0.02, 0.09 \rangle\|, \\ 0, \\ 0.005, \\ \|o_i - h_i + \langle 0.05, 0.02, 0.09 \rangle\| - 0.005))) \end{aligned}$$

E.1.38 Bin Picking

Two funnel-shaped surfaces guide the gripper as it seeks to carry the object between the two bins; this prevents the gripper from running into the side of the bins. The height (or "altitude") of this surface is given by alt since the variables h and z are already used.

$$\begin{aligned} alt &= \min(\mathbb{I}_{\|h_{(xy)} - o_{i,(xy)}\| > 0.03} \cdot (0.2 + 0.02 \log (\|h_{(xy)} - o_{i,(xy)}\| - 0.03)), \\ &\quad \mathbb{I}_{\|h_{(xy)} - t_{(xy)}\| > 0.03} \cdot (0.2 + 0.02 \log (\|h_{(xy)} - t_{(xy)}\| - 0.03))) \\ A &= \mathbb{I}_{\|o - h\| < 0.04 \& g < 0.43 \& o_{(z)} - o_{i,(z)} > 0.02} \cdot \left(1 + 5T_{H_0} \left(\begin{array}{l} L(\|t - o\|, 0, 0.05, \|t - o_i\|), \\ \max(\mathbb{I}_{h_{(z)} > alt}, L(alt - h_{(z)}, 0, 0.01, 0.05)) \end{array} \right) \right) \\ R &= \begin{cases} A + T_{H_0} (R_{cage,dense}(0.015, 0.05, 0.01), L(\|t - o\|, 0, 0.05, \|t - o_i\|)) & \|t - o\| > 0.05 \\ 10 & \text{otherwise} \end{cases} \end{aligned}$$

E.1.39 Assembly

In addition to the components described below, the assembly reward is weighted by how level the object is (tilted object quaternions are penalized).

$$\begin{aligned} alt &= \mathbb{I}_{\|t_{(xy)} - o_{(xy)}\| > 0.02} \cdot (0.4 + 0.04 \log (\|t_{(xy)} - o_{(xy)}\| - 0.02)) \\ A &= 0.1 \mathbb{I}_{o_{(z)} > 0.02 \text{ or } \|t_{(xy)} - o_{(xy)}\| < 0.02} + 0.9L(\langle 1, 1, 3 \rangle \langle t_{(x)} - o_{(x)}, t_{(y)} - o_{(y)}, alt - o_{(z)} \rangle, 0, 0.02, 0.4) \\ R &= \begin{cases} 2R_{cage,dense}(0.015, 0.02, 0.01) + 8A & |t_{(x)} - o_{(x)}| > 0.02 \\ 10 & \text{otherwise} \end{cases} \end{aligned}$$

E.1.40 Disassemble

In addition to the components described below, the disassemble reward is weighted by how level the object is (tilted object quaternions are penalized).

$$R = \begin{cases} 2R_{cage,dense}(0.015, 0.02, 0.01) + 6 (0.1 \mathbb{I}_{o_{(z)} > 0.02} + 0.9L(\|t - o\|, 0, 0.02, 0.2)) & o_{(z)} > t_{(z)} \\ 10 & \text{otherwise} \end{cases}$$

E.1.41 Hammer

In addition to the components described below, the hammer reward is weighted by how level the object is (tilted object quaternions are penalized).

$$R = \begin{cases} 2R_{cage,dense}(0.015, 0.02, 0.01) + 6 (0.1 \mathbb{I}_{o_{(z)} > 0.02} + 0.9L(\|t - o\|, 0, 0.02, 0.2)) & |o_{(y)} - o_{i,(y)}| > 0.09 \\ 10 & \text{otherwise} \end{cases}$$

E.1.42 Lever Pull

$$R = 10T_{H_0}(L(\|t - o\|, 0, 0.04, \|t - o_i\|), L(\langle 4, 1, 4 \rangle \cdot (h - o + \langle 0, 0.055, 0.07 \rangle), 0, 0.02, \langle 4, 1, 4 \rangle \cdot (h_i - o_i + \langle 0, 0.055, 0.07 \rangle)))$$

E.1.43 Stick Push

Note: a is the second object in the environment, which in this case is a thermos.

$$R = \begin{cases} 2 + 5L(\|t - o\|, 0, 0.12, \|t - o_i\| - 0.12) & \|h - o\| < 0.02, g > 0, o_{(z)} - o_{i,(z)} > 0.01, \|t - a\| > 0.12 \\ + 3L(\|t - a\|, 0, 0.12, \|t - a_i\| - 0.12) & \|h - o\| < 0.02, g > 0, o_{(z)} - o_{i,(z)} > 0.01, \|t - a\| \leq 0.12 \\ 10 & \text{otherwise} \\ R_{cage,dense}(0.04, 0.05, 0.01) & \text{otherwise} \end{cases}$$

E.1.44 Stick Pull

Note: a is the second object in the environment, which in this case is a thermos. in is a condition involving lots of vector offsets from the object observations. It indicates whether the stick is inserted into the thermos' handle or not. The variable *stick_in_place*, and *stick_grabbed* have also been defined so that the reward function fits on one page.

$$stick_in_place = L(\|(o - a) \cdot \langle 1, 1, 2 \rangle\|, 0, 0.12, \|(o_i - a_i) \cdot \langle 1, 1, 2 \rangle\|)$$

$$stick_grabbed = \|h - o\| < 0.02, g > 0, o_{(z)} - o_{i,(z)} > 0.01$$

$$R = \begin{cases} 1 + 6 \cdot stick_in_place & stick_grabbed, \neg in, \|t - a\| > 0.12 \\ 6 + stick_in_place + 2L(\|t - o\|, 0, 0.12, \|t - o_i\|) & stick_grabbed, in, \|t - a\| > 0.12 \\ + L(\|t - a\|, 0, 0.12, \|t - a_i\|) & stick_grabbed, in, \|t - a\| \leq 0.12 \\ 10 & otherwise \\ T_{H_0}(R_{cage,dense}(0.014, 0.05, 0.01), stick_in_place) & otherwise \end{cases}$$

E.1.45 Shelf Place

In addition to the components described below, the shelf-place reward includes negative components that help avoid collision with the shelf.

$$A = \mathbb{I}_{\|o - h\| < 0.025 \& g > 0 \& o_{(z)} > 0.01} \cdot (1 + 5L(\|t - o\|, 0, 0.05, \|t - o_i\|))$$

$$R = \begin{cases} A + T_{H_0}(R_{cage}(0.01, 0.02, 0.05, 0.01), L(\|t - o\|, 0, 0.05, \|t - o_i\|)) & \|t - o\| \geq 0.05 \\ 10 & otherwise \end{cases}$$

E.1.46 Peg Insert

In addition to the components described below, the peg-insert reward includes negative components that help avoid collision with the hole/box into which the peg gets inserted.

$$A = \mathbb{I}_{\|o - h\| < 0.08 \& g > 0 \& o_{(z)} > 0.01} \cdot (1 + 5L(\|\langle 2, 2, 1 \rangle \cdot (t - o)\|, 0, 0.05, \|\langle 2, 2, 1 \rangle \cdot (t - o_i)\|))$$

$$R = \begin{cases} A + T_{H_0}(R_{cage}(0.0075, 0.01, 0.03, 0.005), \\ L(\|\langle 2, 2, 1 \rangle \cdot (t - o)\|, 0, 0.05, \|\langle 2, 2, 1 \rangle \cdot (t - o_i)\|)) & \|\langle 2, 2, 1 \rangle \cdot (t - o)\| \geq 0.07 \\ 10 & otherwise \end{cases}$$

E.1.47 Peg Unplug

$$A = \mathbb{I}_{\|o - h\| < 0.035 \& g > 0.5 \& o_{(x)} - o_{i,(x)} > 0.015} \cdot (1 + 5L(\|t - o\|, 0, 0.05, \|t - o_i\|))$$

$$R = \begin{cases} A + 2R_{cage}(0.01, 0.025, 0.05, 0.005) & \|t - o\| \geq 0.07 \\ 10 & otherwise \end{cases}$$

E.1.48 Soccer

In addition to the components described below, the soccer reward function includes parameters to fine-tune movements near the goal line.

$$R = \begin{cases} 3R_{cage}(0.013, 0.023, 0.05, 0.005) & \|\langle 3, 1, 1 \rangle(t - o)\| \geq 0.07 \\ + 6.5L(\|\langle 3, 1, 1 \rangle(t - o)\|, 0, 0.07, \|\langle 3, 1, 1 \rangle(t - o_i)\|) & otherwise \\ 10 & otherwise \end{cases}$$

E.1.49 Pick Place Wall

The pick-place-wall reward is essentially two pick-place rewards stacked on top of one another. The first pick-place reward incentivizes movement to a neutral midpoint above the wall (to avoid running into it). The second pick-place reward incentivizes movement to the target position. The math is such that there is no discontinuity between the two reward components.

E.1.50 Push Wall

The push-wall reward is the same as the pick-place-wall reward, but without incentives to pick up the object. Additionally, the midpoint is configured to be next to the wall (so that policies push the object around the wall) rather than above the wall.

Task	Success Metric
faucet-open	$\mathbb{I}_{\ o-t\ _2 < 0.07}$
sweep	$\mathbb{I}_{\ o-t\ _2 < 0.05}$
pick-out-of-hole	$\mathbb{I}_{\ o-t\ _2 < 0.07}$
faucet-close	$\mathbb{I}_{\ o-t\ _2 < 0.07}$
push	$\mathbb{I}_{\ o-t\ _2 < 0.05}$
stick-push	$\mathbb{I}_{\ o-t\ _2 < 0.12} \text{ and } \mathbb{I}_{grasped(o)}$
coffee-button	$\mathbb{I}_{\ o-t\ _2 < 0.02}$
handle-pull-side	$\mathbb{I}_{\ o-t\ _2 < 0.08}$
basketball	$\mathbb{I}_{\ o-t\ _2 < 0.08}$
stick-pull	$\mathbb{I}_{\ o-t\ _2 < 0.12} \text{ and } \mathbb{I}_{grasped(o)}$
sweep-into	$\mathbb{I}_{\ o-t\ _2 < 0.05}$
disassemble	$\mathbb{I}_{o_z - o_{z initial} > 0.15}$
assembly	$\mathbb{I}_{\ o-t\ _2 < 0.02} \text{ and } \mathbb{I}_{g_z - o_z > 0}$
shelf-place	$\mathbb{I}_{\ o-t\ _2 < 0.07}$
coffee-push	$\mathbb{I}_{\ o-t\ _2 < 0.07}$
handle-press-side	$\mathbb{I}_{\ o-t\ _2 < 0.02}$
hammer	$\mathbb{I}_{nail \ travels \ > 0.09 \ into \ wood \ block}$
plate-slide	$\mathbb{I}_{\ o-t\ _2 < 0.07}$
plate-slide-side	$\mathbb{I}_{\ o-t\ _2 < 0.07}$
button-press-wall	$\mathbb{I}_{\ o-t\ _2 < 0.03}$
handle-press	$\mathbb{I}_{\ o-t\ _2 < 0.02}$
handle-pull	$\mathbb{I}_{\ o-t\ _2 < 0.05}$
soccer	$\mathbb{I}_{\ o-t\ _2 < 0.07}$
plate-slide-back-side	$\mathbb{I}_{\ o-t\ _2 < 0.07}$
plate-slide-back	$\mathbb{I}_{\ o-t\ _2 < 0.07}$
drawer-close	$\mathbb{I}_{\ o-t\ _2 < 0.055}$
reach	$\mathbb{I}_{\ o-t\ _2 < 0.05}$
button-press-topdown-wall	$\mathbb{I}_{\ o-t\ _2 < 0.02}$
reach-wall	$\mathbb{I}_{\ o-t\ _2 < 0.05}$
peg-insert-side	$\mathbb{I}_{\ o-t\ _2 < 0.07}$
push-wall	$\mathbb{I}_{\ o-t\ _2 < 0.07}$
pick-place-wall	$\mathbb{I}_{\ o-t\ _2 < 0.07}$
button-press	$\mathbb{I}_{\ o-t\ _2 < 0.02}$
button-press-topdown	$\mathbb{I}_{\ o-t\ _2 < 0.02}$
pick-place	$\mathbb{I}_{\ o-t\ _2 < 0.07}$
push-back	$\mathbb{I}_{\ o-t\ _2 < 0.07}$
coffee-pull	$\mathbb{I}_{\ o-t\ _2 < 0.07}$
peg-unplug-side	$\mathbb{I}_{\ o-t\ _2 < 0.07}$
dial-turn	$\mathbb{I}_{\ o-t\ _2 < 0.07}$
lever-pull	$\mathbb{I}_{rad(o) - rad(t) < \pi/24}$
window-close	$\mathbb{I}_{\ o-t\ _2 < 0.05}$
window-open	$\mathbb{I}_{\ o-t\ _2 < 0.05}$
door-open	$\mathbb{I}_{\ o-t\ _2 < 0.08}$
door-close	$\mathbb{I}_{\ o-t\ _2 < 0.08}$
drawer-open	$\mathbb{I}_{\ o-t\ _2 < 0.03}$
hand-insert	$\mathbb{I}_{\ o-t\ _2 < 0.05}$
box-close	$\mathbb{I}_{\ o-t\ _2 < 0.08}$
door-lock	$\mathbb{I}_{\ o-t\ _2 < 0.02}$
door-unlock	$\mathbb{I}_{\ o-t\ _2 < 0.02}$
bin-picking	$\mathbb{I}_{\ o-t\ _2 < 0.05}$

Table 12: A list of success metrics used for each of the Meta-World tasks. All units are in meters.