On Generalization and Benchmarking on Physical Robots

Gaoyue Zhou

CMU-RI-TR-23-67

July 21, 2023

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Professor Abhinav Gupta, *chair*
Professor David Held
Sudeep Dasari, *Carnegie Mellon University*

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

To my beloved family.

Abstract

Robot learning research has seen significant advancements. However, the field remains predominantly demo-driven, making direct comparisons between methods difficult without replicating them on individual setups. This places a substantial burden on making scientific progress. While many simulation benchmarks exist, they usually feature contrived datasets and do not accurately reflect real-world performance. The absence of widely recognized benchmarks and real-world applicability makes it difficult to ascertain scientific advancements.

In my thesis, I propose two works that tackle these challenges. In Chapter 2, instead of assuming access to datasets of any quality, we suggest that near-optimal and safe demonstrations collected from out-of-domain tasks are more practical data-sources for real world robot learning. We further propose a set of experiments that evaluate the generalization capabilities of offline reinforcement learning (ORL) and imitation learning methods within this framework. Our study finds that ORL and imitation learning prefer different action spaces, and that ORL algorithms can generalize from leveraging offline heterogeneous data sources and outperform imitation learning.

The second work, introduced in Chapter 3, presents an initiative towards establishing a real-robot benchmark: shared tasks and robots for evaluation that are remotely submitted to and an open-source dataset in this setting. Our benchmark suite includes common manipulation tasks that require challenging generalization to unseen objects, positions, and lighting. Initial results from the benchmark and the launch of a NeurIPS competition highlight the feasibility of such systems.

Acknowledgments

I am immensely grateful to my advisor, Prof. Abhinav Gupta, for his constant support and guidance throughout my Master's journey. His unwavering mentorship has been instrumental in shaping the direction of my research and inspiring me to reach new heights. I also extend my sincere thanks to Prof. David Held and Sudeep Dasari, who served as my committee members. Their valuable feedback and constructive insights have played a crucial part in refining my work, and I won't be able to be where I am without their support. I would also like to thank Vikash Kumar, for his consistent guidance and mentorship.

I would also like to thank Liyiming Ke, Victoria Dean, and Mohan Kumar Srirama for their exceptional collaboration and friendship, enriching my research journey. Furthermore, I am grateful for the camaraderie and support I have found within this vibrant group at CMU. Yufei Ye, Jared Mejia, Homanga Bharadhwaj, Unnat Jain, Jianren Wang, Dingkun Guo, Shikhar Bahl, Raunaq Bhirangi, Sam Powers, Gaurav Parmar, Shun Iwase, Zhizhuo Zhou, Daohan Lu, Jinqi Luo, Tianyuan Zhang, and Xuxin Cheng have all made my time at CMU immensely enjoyable and memorable.

Lastly, but most importantly, I extend my deepest appreciation to my family. Their unwavering love, encouragement, and support have been my rock and foundation throughout every step of my life. I am who I am today because of their unconditional care and belief in me.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

Robot learning has seen great advances over the years. While it is encouraging to see more and more demonstrations on physical robots instead of only in simulation, these demonstrations alone prove inadequate for monitoring scientific advancements since it is often challenging to tell the benefit of one method over the other solely from those demonstrations. The current demo-driven nature of robot learning research makes benchmarking necessary for us to compare methods, identify scientific progress, and make faster improvements.

Nevertheless, the amount of effort in this domain remains insufficient, and robotics as a field struggles to establish a common benchmark. The challenges of establishing a benchmark for robot learning research are manifold. Firstly, consensus has yet to be reached among robot learning researchers regarding the datasets and tasks suitable for benchmarking, stymying initial efforts in this direction. Second, a major goal for robot learning research is the robot's generalization and transfer ability, which is nontrivial to evaluate as an extensive array of scenes and evaluation trials are needed for each model. Furthermore, while simulation-based benchmarks exist and are comparatively simpler to develop and maintain, simulators by far are not a good proxy for real-world performance thus hold limited relevance for real-world robotics. Lastly, a benchmarking effort should encourage accessibility for a broader community, which is a challenge for any centralized setup.

In this thesis, we propose two works that tackle the challenges in establishing benchmarks for robot learning research. We focus on evaluating the generalization and

transfer ability of offline learning algorithms under a realistic data assumption, and propose a benchmark with open source, shared hardware enabling fair comparisons.

In the first work, Real World Offline Reinforcement Learning with Realistic Data Source, we posit that data collected from safe operations of closely related tasks are more practical data sources for real-world robot learning. Under these settings, we perform an extensive (6500+ trajectories collected over 800+ robot hours and 270+ human labor hours) empirical study evaluating generalization and transfer capabilities of representative ORL methods on four real-world tabletop manipulation tasks. Our study finds that ORL and imitation learning prefer different action spaces, and that ORL algorithms can generalize from leveraging offline heterogeneous data sources and outperform imitation learning.

In the second work, Train Offline, Test Online: A Real Robot Learning Benchmark, we propose a new benchmark: Train Offline, Test Online (TOTO). TOTO provides remote users with access to shared robotic hardware for evaluating methods on common tasks and an open-source dataset of these tasks for offline training. Its manipulation task suite requires challenging generalization to unseen objects, positions, and lighting. We present initial results on TOTO comparing five pretrained visual representations and four offline policy learning baselines, remotely contributed by five institutions. The real promise of TOTO, however, lies in the future: we release the benchmark for additional submissions from any user, enabling easy, direct comparison to several methods without the need to obtain hardware or collect data.

Chapter 2

Real World Offline Reinforcement Learning with Realistic Data Source

2.1 Introduction

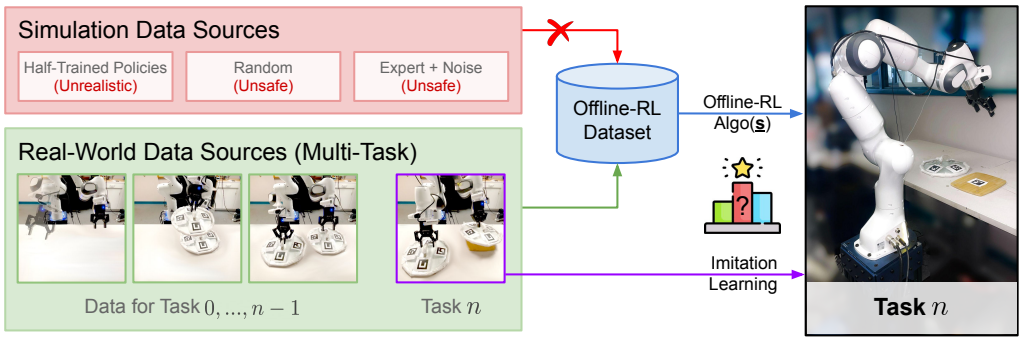


Figure 2.1: Realistic data sources for offline RL algorithms in real world tasks.

Despite rapid advances, the applicability of Deep Reinforcement Learning (DRL) algorithms [28, 49, 54, 68, 70, 71, 76, 81] to real-world robotics tasks is limited due to sample inefficiency and safety considerations. The emerging field of offline reinforcement learning (ORL) [46, 48] has the potential to overcome these challenges, by learning only from logged or pre-generated offline datasets, thereby circumventing

safety and exploration challenges. This makes ORL well suited for applications with large datasets (e.g. recommendation systems) or those where online interactions are scarce and expensive (e.g. robotics). However, comprehensive benchmarking and empirical evaluation of ORL algorithms is significantly lagging behind the burst of algorithmic progress [9, 20, 33, 35, 36, 41, 55, 77, 83, 90, 91]. Widely used ORL benchmarks [18, 25] are entirely in simulation and use contrived data collection protocols that do not capture fundamental considerations of physical robots. Therefore, they hold limited relevance for real-world robotics. In this work (Real-ORL), we aim to bridge this gap by outlining practical offline dataset collection protocols that are representative of real-world robot settings. Our work also performs a comprehensive empirical study spanning **6500+ trajectories collected over 800+ robot hours and 270+ human labor hour**, to benchmark and analyze three representative ORL algorithms thoroughly. We will release all the datasets, code, and hardware hooks from this paper.

In principle, ORL can be used to train policies from datasets of arbitrary quality. This has prompted the development of simulated ORL benchmarks [18, 25, 85] that utilize data sources like expert policies trained with *online* RL, exploratory policies, or even replay buffers of *online* RL agents. However, simulated datasets may fail to capture the challenges in real world: hardware noises coupled with varying reset conditions lead to covariate shift and violate the i.i.d. assumption about state distributions between train and test time. Further, such datasets are not feasible on physical robots and defeat the core motivation of ORL for robotics – to avoid the use of *online* RL due to poor sample efficiency and safety! Recent works [64, 85] suggest that dataset composition and distribution dramatically affect the relative performance of algorithms. In this backdrop, we consider the pertinent question:

What is a practical instantiation of the ORL setting for physical robots, and can existing ORL algorithms learn successful policies in such a setting?

In this work, we envision *practical* scenarios to apply ORL for real-world robotics. Towards this end, our first insight is that real-world offline datasets are likely to come from well-behaved policies that abide by safety and monetary constraints, in sharp contrast to simulator data collected from exploratory or partially trained policies, as used in simulated benchmarks [18, 25, 85]. Such trajectories can be collected by user demonstrations or through hand-scripted policies that are partially successful but

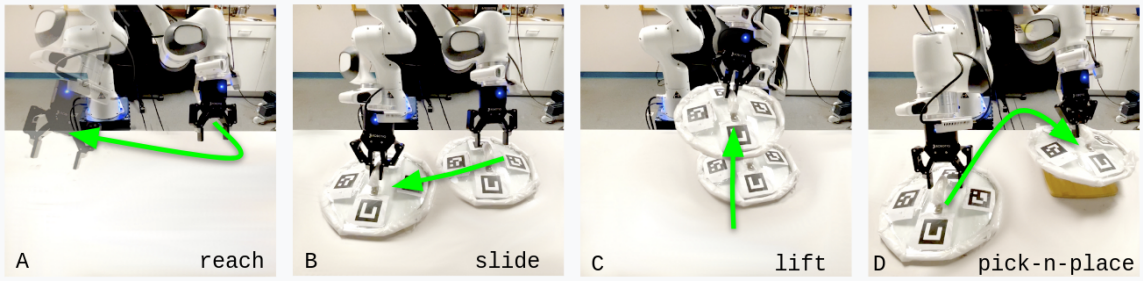


Figure 2.2: Canonical tasks for tabletop manipulation.

safe. It is more realistic to collect large volumes of data for real robots using multiple successful policies designed under expert supervision for specific tasks than using policies that are unsuccessful or without safety guarantees. Secondly, the goal of any learning (including ORL) is broad generalization and transfer. It is therefore critical to study whether a learning algorithm can leverage task-agnostic datasets, or datasets intended for a source task, to make progress on a new target task. In this work, we collect offline datasets consistent with these principles and evaluate representative ORL algorithms on a set of canonical table-top tasks as illustrated in Figure 2.2.

Effort on evaluations conducted on physical robots is sparse in the field due to time and resource constraints, although it is vital to further our understanding. Our real robot results corroborate and validate intuitions from simulated benchmarks [43] but also enable novel discoveries. We find that (1) even for scenarios with sufficiently high-quality data, some ORL algorithms could outperform behavior cloning (BC) [62] on certain tasks, (2) for scenarios that require generalization or transfer to new tasks with low data support, ORL agents generally outperform BC. (3) in cases with overlapping data support, ORL algorithms can leverage additional heterogeneous task-agnostic data to improve their own performance, and in some cases even surpass the best in-domain agent which is trained only from data of the target task.

Our empirical evaluation is unique as it focuses on ORL algorithms ability to leverage more realistic, multi-task data sources, spans over several tasks that are algorithm-agnostic, trains various ORL algorithms on the same settings and evaluates them directly in the real world. In summary, we believe Real-ORL establishes the effectiveness of offline RL algorithms in leveraging out of domain high-quality heterogeneous data for generalization and transfer in robot-learning, which is representative

of real world applications.

2.2 Preliminaries and Related Work

Offline RL. We consider the ORL framework, which models the environment as a Markov Decision Process (MDP): $M = \langle S, A, R, T, \rho_0, H \rangle$ where $S \subseteq \mathbb{R}^n$ is the state space, $A \subseteq \mathbb{R}^m$ is the action space, $R : S \times A \to \mathbb{R}$ is the reward function, $T : S \times A \times S \to \mathbb{R}_+$ is the (stochastic) transition dynamics, $\rho_0 : S \to \mathbb{R}_+$ is the initial state distribution, and H is the maximum trajectory horizon. In the ORL setting, we assume access to the reward function R and a pre-generated dataset of the form: $\mathbb{D} = \{\tau_1, \tau_2, \ldots \tau_N\}$, where each $\tau_i = (s_0, a_0, s_1, a_1, \ldots s_H)$ is a trajectory collected using a behavioral policy or a mix of policies $\pi_b : S \times A \to \mathbb{R}_+$.

The goal in ORL is to use the offline dataset $\mathbb{D}$ to learn a near-optimal policy,

$$\pi^* := \arg\max_{\pi} \; \mathbb{E}_{M,\pi} \left[\sum_{t=0}^{H} r(s_t, a_t) \right].$$

In the general case, the optimal policy π^* may not be learnable using $\mathbb{D}$ due to a lack of sufficient exploration in the dataset. In this case, we would seek the best policy learnable from the dataset, or, at the very least, a policy that improves upon behavioral policy.

Offline RL Algorithms. Recent years have seen tremendous interests in offline RL and the development of new ORL algorithms. Most of these algorithms incorporate some form of regularization or conservatism. This can take many forms, such as regularized policy gradients or actor critic algorithms [9, 19, 29, 55, 83, 84], approximate dynamic programming [20, 36, 41, 77], and model-based RL [1, 33, 87, 88]. We select a representative ORL algorithms from each category: AWAC [55], IQL [36] and MOReL [33]. In this work, we do not propose new algorithms for offline RL; rather we study a spectrum of representative ORL algorithms and evaluate their assumptions and effectiveness on a physical robot under realistic usage scenarios.

Offline RL Benchmarks and Evaluation. In conjunction with algorithmic advances, offline RL benchmarks have also been proposed. However, they are predominantly captured with simulation [18, 25, 44] using datasets with idealistic coverage, i.i.d. samples, and synchronous execution. Most of these assumptions are invalid in real world which is stochastic and has operational delays. Prior works investigating offline RL for these settings on physical robots are limited. For instance, Kostrikov et al. [36] did not provide real robot evaluation for IQL, which we conduct in this work; Chebotar et al. [8], Kalashnikov et al. [31] evaluate performance on a specialized Arm-Farm; Rafailov et al. [66] evaluate on a single drawer closing task; Kumar et al. [42], Singh et al. [77] evaluate only one algorithm (COG, CQL, respectively). Mandlekar et al. [52] evaluate BCQ and CQL alongside BC on three real robotics tasks, but their evaluations consider only in-domain setting: that the agents were trained only on the specific task data, without giving them access to a pre-generated, offline dataset. Thus, it is unclear whether insights from simulated benchmarks or limited hardware evaluation can generalize broadly. Our work aims to bridge this gap by empirically studying representative offline RL algorithms on a suite of real-world robot learning tasks with an emphasize on transfer learning and out-domain generalization. See Section 2.3 for detailed discussion.

Imitation Learning (IL). IL [57] is an alternate approach to training control policies for robotics. Unlike RL, which learns policies by optimizing rewards (or costs), IL (and inverse RL [30, 34, 93]) learns by mimicking expert demonstrations and typically requires no reward function. IL has been studied in both the offline setting [32, 89], where the agent learns from a fixed set of expert demonstrations, and the online setting [7, 67], where the agent can perform additional environment interactions. A combination of RL and IL has also been explored in prior work [69, 78]. Our offline dataset consists of trajectories from a heuristic hand-scripted policy collected under expert supervision, which represents a dataset of reasonably *high quality*. As a result, we consider offline IL and, behavior cloning in particular, as a baseline algorithm in our empirical evaluation.

2.3 Experiment Scope and Setup

To investigate the effectiveness of ORL algorithms on real-world robot learning tasks, we adhere to a few guiding principles: (1) we make design choices representing the wider community to the extent possible, (2) we strive to be fair to all baselines by providing them their best chance and work in consultation with their authors; and (3) we prioritize reproducibility and data sharing. We have open-sourced our data, camera images along with our training and evaluation codebase.

Hardware Setup. Hardware plays a seminal role in robotic capability. For reproducibility and extensibility, we selected a hardware platform that is well-established, non-custom, and commonly used in the field. After an exhaustive literature survey [3, 17, 24, 38, 56, 75], we converged on a table-top manipulation setup shown in Figure 2.3. It consists of a table-mounted Franka panda arm that uses a RobotiQ parallel gripper as its end effector, which is accompanied by two Intel 435 RGBD cameras. Our robot has 8 DOF, uses factory-supplied default controller gains, accepts position commands at 15 Hz, and runs a low-level joint position controller at 1000 Hz. To perceive the object to interact with, we extract the position of the AprilTags attached to the object from RGB images. Our robot states consist of joint positions, joint velocities, and positions of the object to interact with (if applicable). Our policies compute actions (desired joint positions) using robot proprioception, tracked object locations, and desired goal location.

Canonical Tasks We consider four classic manipulation tasks common in literature: `reach`, `slide`, `lift`, and `pick-n-place (PnP)` (see Figure 2.2). `reach` requires the robot to move from a randomly sampled configuration in the workspace to another configuration. The other three tasks involve a heavy glass lid with a handle, which is initialized randomly on the table. `slide` requires the robot to hold and move the lid along the

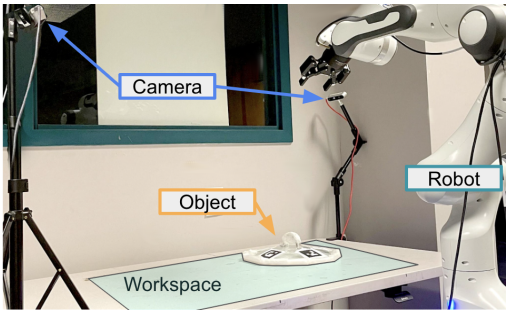


Figure 2.3: Our setup consists of a commonly used Franka arm, a RobotiQ parallel gripper, and two Intel Realsense 435 cameras.

table to a specified goal location. `lift` requires the robot to grasp and lift the lid 10 cm off the table. `PnP` requires the robot to grasp, lift, move and place the lid at a designated goal position i.e. the chopping board. The four tasks constitute a representative range of common tabletop manipulation challenges: `reach` focuses on free movements while the other three tasks involve intermittent interaction dynamics between the table, lid, and the parallel grippers. We model each canonical task as a MDP with an unique reward function. Details on our tasks are in Appendix A.1.

Data Collection. We use a hand-designed, scripted policy developed under expert supervision to collect (dominantly) successful trajectories for all our canonical tasks. To highlight ORL algorithms ability to overcome suboptimal dataset, previous works [18, 44, 52] have crippled expert policies with noise, use half-trained RL policies or collect human demonstrations with varying qualities to highlight the performance gain over compromised datasets. We posit that such data sources are not representative of robotics domains, where noisy or random behaviors are unsafe and detrimental to hardware's stability. Instead of infusing noise or failure data points to serve as negative examples, we believe that mixing data collected from various tasks offers a more realistic setting in which to apply ORL on real robots for three reasons: (1) collecting such "random/roaming/explorative" data on a real robot autonomously would require comprehensive safety constraints, expert supervision and oversight, (2) engaging experts to gather extensive quantities of random data is less logical than employing their expertise to collect meaningful trajectories from an actual task, and (3) designing task-specific strategies and stress testing ORL's ability against such a strong dataset is more viable than using a compromised dataset. In Real-ORL, we collected offline dataset using heuristic strategies designed with reasonable efforts and froze the dataset ahead of time to avoid biases favoring certain tasks and algorithms.

To implement scripted policies for all tasks, we first decompose each task into simpler stages marked by end-effector sub-goals. We leverage Mujoco's IK solver to map these sub-goals into joint space. The scripted policy takes tiny steps toward sub-goals until some task-specific criteria are met. Our heuristic policies didn't reach the theoretical maximum possible scores due to controller noises and tracking noises (Table 2.1). However, they complete the task at a high success rate and have comparable performance to human demonstration. More information of the

performance of the dataset can been seen in Table 2.1.

Dataset. Our Real-ORL dataset consists of around 3000 trajectories and the characteristics of our dataset is shown in Table 2.1. Our offline dataset is represented as a series of transition tuples $\{(s, a, s')_{task}\}$. States consist of joint positions, joint velocities, and positions of the object to interact with (if applicable). Actions contain target joint positions. To perceive the object to interact with, we obtain the position of tracked AprilTags attached to the object from the RGB images of the two cameras. More details are available in Appendix A.2.

Table 2.1: Characteristics of collected data. *# Traj* denotes the total number of trajectories, *# Samples* denotes the total number of state-action-reward pairs. Each trajectory's score is the maximum reward in the trajectory. *Avg Score* shows the average scores per trajectories, *Max Score* shows the maximum reward achieved by trajectories in our dataset, *Human Score* shows the max reward achieved by a human teleoperator and *Theoretical Best Score* denotes the theoretical maximum possible reward determined by our reward function.

Task	*# Traj*	*# Samples*	*Avg Score*	*Max Score*	*Human Score*	*Theoretical Best Score*
`reach`	1000	99752	0.960	0.99	0.963	1
`slide`	731	244422	0.819	0.93	0.834	1
`lift`	609	178515	0.948	1	1	1
`PnP`	616	327478	0.875	1.09	0.924	1.15

2.4 Experiment Design

Our Real-ORL experiments aim to answer the following questions:

1. Are ORL algorithms sensitive to, or show a preference for, any specific state and action space parameterization?
2. How do they perform against the standard methods for in-domain tasks?
3. How do common methods perform in out-of-domain tasks requiring (a) generalization, and (b) re-targeting?

To ensure fair evaluation, we now outline our choice of candidate algorithms and performance metrics.

Algorithms For all evaluations, we compare four algorithms: Behavior Cloning (BC) [62], Model-based Offline REinforcement Learning (MOReL) [33], Advantage-Weighted Actor Critic (AWAC) [55] and Implicit Q-Learning (IQL) [36]. BC is a model-free IL algorithm that remains a strong baseline for real robot experiments due to its simplicity and practicality. AWAC and IQL both train an off-policy value function and then derive a policy to maximize the expected reward. AWAC uses KL divergence minimization to constrain the resulting policy to be close to the given policy distribution. In contrast, IQL leverages expectile regression to avoid querying the value function for any out-of-distribution query. MOReL is distinct since it is a model-based approach: it recovers a dynamics model from offline data that allows it directly apply policy gradient RL algorithms. We use implementations of BC and MOReL from the MOReL author implementation. For the later, we add a weighted behavior cloning loss to its policy training step to serve as a regularizer, inspired by [19]. We use AWAC and IQL implemented in the open sourced d3rlpy library [73].

Training. Since neural network agents are empirically sensitive to parameters and seeds, we (1) used the same fixed random seed (123) for all our experiments with additional seed sweeping to strengthen the reproducibility of our results and (2) conducted equal amount of efforts for hyperparameter tuning efforts for all algorithms. Unlike traditional supervised learning, we cannot simply select the agents with the best validation loss for tuning the hyperparameters, because we donnot know the performance of an agent unless testing it on a real robot [52]. We thus keep our tuning simple and fair: starting with the default parameters and training 5 agents in 3 rounds, trying to make the agent converge. We observe that certain agents cannot converge after exhausting the allocated trials and report these results with a (*) marker, signaling the challenge in tuning parameters for such algorithms. More details are available in Appendix A.4.

Evaluation. Real robot evaluations can have high variance due to reset conditions and hardware noise. For each agent, we collect 12 trajectories and report their mean and standard deviation of scores. To confirm the reproducibility of our results and robustness to seed sweeping, for agents that contributed to our conclusions (usually the best and the second-best agents) we report performance swept over three consecutive

random seeds (122, 124 in addition to the fixed seed of 123) in Appendix A.6. To verify the statistical significance of our results when comparing performance between agents, we report the p-value of paired difference tests in Appendix A.7.

To answer the questions mentioned at the beginning of this section, we design two sets of real-world experiments: in-domain and out-domain experiments. We note the distinction between "in-domain" training and "out-domain" training, where the former leverages only data that were collected for the test task and the later allow incorporating heterogeneous data from different tasks.

A. In-domain Ablations. We train all agents using in-domain data (i.e., we train a `slide` agent by feeding only `slide` data) to test ORL algorithms' sensitivity to varying data representation and inspect: (1) whether it is worth including velocity information in the state space (**Vel** versus **NoVel**); for simulator experiments, it is almost always a gain to include velocity, but velocity sensors on real robots are notoriously noisy; (2) whether to use the policy output joint position (**Abs**) vs the change in joint position (**Delta**) as action. Most BC literature uses the former, whereas RL prefers the latter. [1] We use the outcome of the ablations and the best-performing setting for each algorithm to study generalization and transfer in the following three scenarios:

B. Generalization: Lacking data support. Regardless of the data collection method employed, the coverage of the task space may not be uniform. For example, imagine that a robot trained to wipe clean a table but now cleans a bigger table. Empirically, a policy trained with behavior cloning would have trouble predicting actions for states when there is less data support due to the supervised nature of behavior cloning. Can ORL algorithms, by learning a value function and perhaps a dynamics model, generalize to a task space that lacks data support? To this end, we create a new dataset from our `slide` task by dividing the task space to three regions: left, center, right. We remove any trajectory where the object was initially placed in

[1] Additionally, to verify that our dataset has reasonable optimality sufficient for training BC, we train BC separately with Top-K% of the trajectories to exclude the relatively "worse" trajectories. The results showns in Appendix. A.5 verifies that BC has the best performance using the full dataset we collect.

the center region from the collected dataset. We train all agents and gather evaluation trajectories asking them to slide an object initially placed in the left, center and right regions.

C. Generalization: Re-targeting data for dynamic tasks. For the `slide` task, our collected demonstration has static goal positions. We test agents trained using such static data in a dynamic setting by updating the goal at a fixed frequency, and asking the agents to grasp and slide the lid following some predetermined curves. We collected the ideal trajectories via human demonstration. This task can be viewed as a simplified version of daily tasks, including drawing, wiping, and cleaning, which require possibly repeated actions and a much longer horizon than usual IL and ORL tasks. We select a variety of trajectories: circle, square, and the numbers 3, 5, 6, 8, which have different combinations of smooth curves and corners.

D. Transfer: Reusing data from different tasks. We investigate whether we can reuse hetergenuous data collected from previous tasks to train a policy for a new task. For example, would combining data from two canonical tasks (e.g., `slide`+`lift`) helps the agent perform better on either of these tasks? The ability for robot learning to utilize datasets from various tasks holds significant importance as it allows robots to acquire adaptable and versatile skills from existing resources. When aggregating data collected for multiple tasks, ORL algorithms can use the reward function for the test task to relabel the offline dataset. Evaluating ORL algorithms on such out-domain, transfer-learning settings is practical and relevant: instead of collecting random explorative data which demands careful setup of safety constraints on a real robot, we want to leverage offline datasets collected from different tasks to improve ORL performance. We train our algorithm with different combinations of canonical task demonstrations ("train-data") and evaluate each agent on each individual task.

2.5 Results and Discussion

2.5.1 In-domain Tasks

Which agent performs best for in-domain tasks? Table 2.2 summarizes all agents' performance for *in-domain* tasks. The bottom row of the table plots the average reward across all four tasks for each agent. Interestingly, two of the ORL agents, IQL and AWAC, achieved higher scores than BC trained on 2 out of 4 tasks. After verifying statistical significance, we confirm that even with abundant, in-domain demonstrations, IQL outperformed BC on two tasks. For the other tasks: on the simplest task `reach`, the best version of all agents reached comparable performance. On the hardest task `PnP`, BC outperformed the best ORL agents. We thus recommend considering both BC and IQL as baselines for in-domain robot learning tasks which favor imitation learning.

Sensitivity to representation Empirically, BC demonstrated robustness to different state and action spaces, whereas ORL agents had high-variance. In 3 of 4 tasks considered, BC performed the best when using absolution joint position as the action space and including velocity in the state space (**AbsVel**). On all tasks, ORL agents performed better using delta action space (**Delta**) rather than joint position. Intuitively, using the delta action space would be equivalent to restricting the policy to move in a unit ball centered around the current state. Such constraints could benefit RL policies which need exploration and sampling in action space more than it helped BC, which simply learns the mapping from states to actions. We also observed that our best agents all included velocity in their state space, despite that velocities on real robot fluctuate with hardware noise.

2.5.2 Generalization and Transfer

Generalization to regions that lack data support. Table 2.3 trains agent using a carved-out dataset and compares the agents' performance on regions with more data support versus the region with less data support (`Center`). We also train all agents using the full dataset (without carved-out) and evaluate them on

Table 2.2: Performance of all algorithms on varying representations. Each agent for each task is trained and evaluated on four settings: to include velocity in state or not (**Vel** versus **NoVel**); to use absolute or delta action space (**Abs** versus **Delta**). For each task, the best BC agent and the best ORL agent are highlighted and bolded. Agents that could not converge during training time are marked with (*). Some agents triggered violent crashes at test time and we report such performance as <0. Underline scores are swept over 3 seeds.

Task	**Agent**	Representations			
		AbsNoVel	**AbsVel**	**DeltaNoVel**	**DeltaVel**
Reach	BC	0.863 ± 0.069	0.768 ± 0.118	0.912 ± 0.026	**0.924 ± 0.048**
	Morel	0.795 ± 0.086	0.584 ± 0.105	0.86 ± 0.069	0.917 ± 0.036
	AWAC	0.770 ± 0.105	0.713 ± 0.158	0.916 ± 0.030	**0.925 ± 0.047**
	IQL	0.843 ± 0.148	0.872 ± 0.104	0.904 ± 0.032	0.894 ± 0.066
Slide	BC	0.623 ± 0.172	**0.681 ± 0.147**	0.548 ± 0.200	0.551 ± 0.101
	Morel	0.356 ± 0.189	0.117 ± 0.235	0.532 ± 0.147	0.629 ± 0.160
	AWAC	0.548 ± 0.171 *	0.591 ± 0.146 *	0.569 ± 0.138	0.732 ± 0.113
	IQL	0.627 ± 0.144	0.589 ± 0.166	0.712 ± 0.137	**0.767 ± 0.065**
Lift	BC	0.759 ± 0.179	**0.823 ± 0.177**	0.721 ± 0.225	0.613 ± 0.142
	Morel	0.460 ± 0.189	0.149 ± 0.092	0.678 ± 0.186	0.652 ± 0.160
	AWAC	0.518 ± 0.083 *	<0 *	0.863 ± 0.149 *	0.821 ± 0.121
	IQL	0.682 ± 0.163	<0	0.841 ± 0.144	**0.880 ± 0.149**
PnP	BC	0.632 ± 0.123	**0.818 ± 0.185**	0.564 ± 0.045	0.678 ± 0.195
	Morel	<0	<0	**0.750 ± 0.197**	0.748 ± 0.220
	AWAC	0.451 ± 0.159 *	<0	0.626 ± 0.234 *	0.735 ± 0.175 *
	IQL	0.469 ± 0.142	<0	0.548 ± 0.160	0.601 ± 0.228

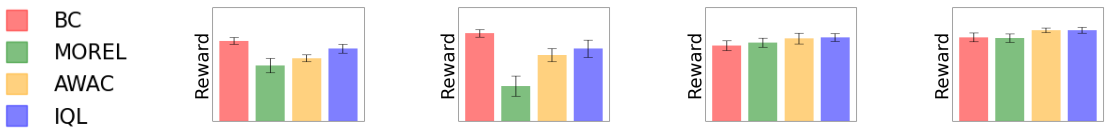


Table 2.3: Training agents using a carved-out dataset to see how they perform when *generalizing* to a task region that lacks data support (the `Center` region, highlighted in Gray). For comparison, we also train all agents using full dataset and evaluate them on the `Center` region.

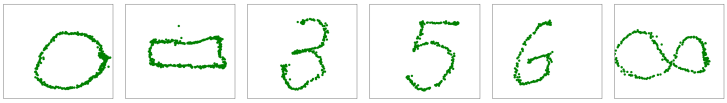
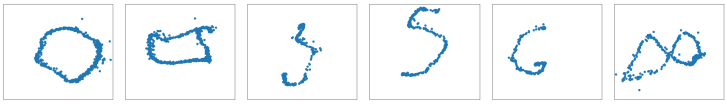
Start Position	**BC**	**MOReL**	**AWAC**	**IQL**
`Left`	0.790 ± 0.056	0.571 ± 0.062	0.704 ± 0.119	0.704 ± 0.066
`Right`	0.774 ± 0.015	0.799 ± 0.033	0.707 ± 0.136	0.808 ± 0.015
`Center`	0.764 ± 0.013	0.793 ± 0.015	0.830 ± 0.026	0.811 ± 0.007
`Center`, trained with full data	0.791 ± 0.018	0.776 ± 0.022	0.813 ± 0.021	0.811 ± 0.050

the `Center` region. We discovered that: (1) on the region with abundant support (`Left` and `Right`), BC/IQL performed better than AWAC/MOREL , aligning with our previous observation that BC/IQL performed better on in-domain tasks, (2) on regions that have less data support, AWAC and MOREL could match BC's performance despite their initial disadvantage; and (3) ORL agents trained with carved-out dataset and evaluated on carved-out region performed no worse than them trained with full dataset, in contrast to BC agent, which performed significantly worse after carving-out.

Generalization to dynamic tasks. Table 2.4 lists the ideal curves (collected via human teleoperation) and the curves traced by each model. Each dot represents the location of the lid at a certain time step. BC had the worst performance among all models since it failed to complete tracing of the circle, square, and number 8 which requires a larger range of motion, and the BC agent seemed to get stuck during execution. Meanwhile, ORL methods largely succeeded tracing the entire curve following the time-varying goals, demonstrating stronger generalizing ability for this dynamic task.

Transfer learning by leveraging heterogeneous dataset Table 2.5 evaluates the performance of ORL algorithms when trained with different combinations of datasets from multiple tasks. The last column plots the average reward across all three tasks for each dataset combination. We observe that:

Table 2.4: Trajectory tracking. Green: the ideal demo trajectories, followed by each agent's tracking trajectories.

Ideal Traj						
BC						
MOREL						
AWAC						
IQL						

1. The change in performance of ORL agents after leveraging offline data from different tasks can vary, due to the characteristics of the algorithm, the nature of the task, design of the reward function and the data distribution.

2. We observed that all ORL agents *could* improve their own performance using some task/data combinations. Noticeably, MOREL achieved higher or comparable performance on all tasks after leveraging more offline data. For instance, its performance on the `lift` task progressively improved ($0.606 \rightarrow 0.726 \rightarrow 0.896$) with the inclusion of data from `slide` and `PnP` tasks. Intuitively, MOREL's dynamic model training process could benefit from any realistic data, regardless of whether the data was in-domain or out-of-domain.

3. Certain task-agnostic data could provide overlapping data support and enable effective transfer learning, allowing some ORL agents to surpass imitation learning and even the best in-domain agents. On `slide` and `lift`, all ORL algorithms managed to surpass BC. On `PnP`, AWAC achieved comparable performance as BC but with a slightly higher mean using a combo of `slide` and `lift` data. With our extensive ablations, we observe that the final best agent for each task is either an ORL algorithm or a tie between ORL and BC.

4. ORL algorithms are not guaranteed to increase performance by including more data. The performance changes of ORL are likely to vary by agents, the task and dataset distribution. For instance, both AWAC and IQL agents have not gained performance on `lift` when using `slide`+`lift`+`PnP` than using only `slide`+`lift` data ($0.899 \rightarrow 0.728$, $0.863 \rightarrow 0.684$). Surprisingly, training IQL for `PnP` using `slide` or `slide`+`lift` data yielded even better results than using `PnP` data ($0.84 > 0.6$). Qualitatively we observe that IQL agents trained with `slide` data were better at grasping the object than the ones trained with `PnP` data, completing this first part of the task (grasp) with more success while claiming distance-to-goal reward bonus.

Random Seed Sweeping To further demonstrate the reproducibility of our results, we conducted random seed sweeping for our best and (optionally) the second-best agents over 3 consecutive random seeds ($122, 124$ in addition to the original fixed seed 123) and reported their scores using an underline in Appendix. A.6. These additional 360 trajectories showed that the seed2seed variation for our experiments is low, providing statistical significance to our observations: comparing with scores computed from a single-seed, $\sim$60% of newly trained agents change score by less than 1%, $\sim$90% of agents change by less than 2%, and the maximum change was 6% from one agent (whose score change does not affect the conclusion drawn).

Comparison to previous works. Some of our *in-domain* conclusions are aligned with [52, 57]: that BC demonstrates strong robustness to varying representations and tasks, serving as a competitive baseline in all four tasks tested. Even when BC is not the best, it has reasonable performance that is no worse than 85% of the best in-domain agents. Our findings also provide empirical verification to one of [44]'s observations: that ORL could outperform BC for tasks where the initial state distributions change during deployment, a common condition for real robotic task, or when the environment has a few "critical" states, as seen in our manipulation tasks.

In contrast to previous works, however, we highlight that (1) IQL can be a competitive baseline for settings that were traditionally favoring behavior cloning, as it turns out to be the best in-domain agent on 2 out of 4 tasks we tested, despite the lack of real robotic evaluation for IQL [36], (2) our extensive ablations on out-domain transfer learning are unique and allow us to verify several ORL algorithms' capability

Table 2.5: Performance of agents trained with different combinations of offline data. The best in-domain agent, transfer learning agents that improves over their in-domain counterparts are colored. The **best agent for each task** is bold. Agents that could not converge during training time are marked with (*). Some agents triggered violent crashes at test time and we report such performance as <0. Underline scores are swept over 3 seeds.

Agent	Train Data	Test Task `slide`	Test Task `lift`	Test Task `PnP`	slide / slide+lift / slide+lift+PnP
BC	in-domain	0.681 ± 0.147	0.823 ± 0.177	0.818 ± 0.185	
	slide	0.681 ± 0.147	0.582 ± 0.058	0.612 ± 0.083	
	slide+lift	0.595 ± 0.127	0.580 ± 0.053	0.605 ± 0.120	
	slide+lift+PnP	0.610 ± 0.137	0.609 ± 0.079	0.640 ± 0.144	
MOReL	in-domain	0.629 ± 0.160	0.678 ± 0.186	0.750 ± 0.197	
	slide	0.629 ± 0.160	0.606 ± 0.063	0.744 ± 0.174	
	slide+lift	0.616 ± 0.146	0.726 ± 0.184	0.636 ± 0.173	
	slide+lift+PnP	0.715 ± 0.134	**0.896 ± 0.133**	0.753 ± 0.181	
AWAC	in-domain	0.732 ± 0.113	0.863 ± 0.149 *	0.735 ± 0.175 *	
	slide	0.732 ± 0.113	0.638 ± 0.055	0.770 ± 0.111*	
	slide+lift	0.734 ± 0.110 *	0.899 ± 0.149	0.813 ± 0.121	
	slide+lift+PnP	0.644 ± 0.144 *	0.728 ± 0.200 *	0.758 ± 0.188 *	
IQL	in-domain	**0.767 ± 0.065**	0.880 ± 0.149	0.601 ± 0.228	
	slide	0.767 ± 0.065	0.258 ± 0.033	0.810 ± 0.107	
	slide+lift	0.704 ± 0.141	0.863 ± 0.166	**0.842 ± 0.114**	
	slide+lift+PnP	0.643 ± 0.143	0.684 ± 0.158	0.833 ± 0.183	

in generalizing to task region with less data-support (Table. 2.3) and to dynamic tasks (Figure. 2.4), (3) we observe that leveraging heterogeneous data has enabled all ORL algorithms to improve their own performance on at least one of the tasks, allowing some to even surpass the best in-domain agents, which suggest that ORL can be an interesting paradigm for real-world robotic learning.

2.6 Conclusion

In Real-ORL, we conducted an empirical study of representative ORL algorithms on real-world robotic learning tasks, with the design of four experiments that test for the generalization and transfer ability of ORL and IL algorithms in realistic scenarios. The study encompassed three representative ORL algorithms (along with behavior cloning), four table-top manipulation tasks with a Franka-Panda robot arm, 3000+ train trajectories, 3500+ evaluation trajectories, and 270+ human labor hours. Through our extensive ablation studies, we find that (1) even for in-domain tasks with abundant amount of high-quality data, IQL can be a competitive baseline against the best behavior cloning policy, (2) for out-domain tasks, ORL algorithms were able to generalize to task regions with low data-support and to dynamic tasks, (3) the performance changes of ORL after leveraging heterogeneous data are likely to vary by agents, the design of the task, and the characteristics of the data, (4) certain heterogeneous task-agnostic data could provide overlapping data support and enable transfer learning, allowing ORL agents to improve their own performance and, in some cases, even surpass the best in-domain agents. Overall, (5) the best agent for each task is either an ORL algorithm or a tie between ORL and BC. Our rigorous evaluations indicate that even in out-of-domain multi-task data regime, (more realistic in real world setting) offline RL is an effective paradigm to leverage out of domain data.

2.7 Limitations

Our evaluation primarily focuses on three representative ORL algorithms which have shown strong performance in simulated benchmarks. However, ORL is a rapidly

evolving research field, with many different classes of algorithms [9, 19, 63]. Therefore, expanding the scope of tasks, evaluation, and algorithms would offer interesting and valuable future work. Moreover, all ORL algorithms have several hyperparameters that influence their learning. We followed instructions and conventional wisdom in the community to tune parameters, but acknowledge that our experiments do not preclude the possibility that one could obtain better performing agents using a different set of parameters and seeds. Hyperparameter and model selection for offline RL is an emerging research sub-field [42] and progress here would also help advance the applicability of ORL to robot learning. We hope this study and our open-source codebase will facilitate this undertaking.

In the perspective of benchmarking, although this work has provided comprehensive evaluations of four algorithms, its sustainability in the long term may be hindered by relying solely on the resources of a single research group with the continual emergence of new algorithms and methods. In the work introduced in the subsequent chapter (Chapter 3), we present an alternative approach. We propose to use shared hardware and data for evaluating the generalization of both visual and policy learning methods. This approach extends beyond the confines of a single laboratory, inviting contributions and benchmarking from researchers across the community, thus ensuring a more scalable and sustainable evaluation process.

Chapter 3

Train Offline, Test Online: A Real Robot Learning Benchmark

3.1 Introduction

One of the biggest drivers of success in machine learning research is arguably the availability of benchmarks. From GLUE [82] in natural language processing to ImageNet [16] in computer vision, benchmarks have helped identify fundamental advances in many areas. Meanwhile, robotics struggles to establish common benchmarks due to the physical nature of evaluation. The experimental conditions, objects of interest, and hardware vary across labs, often making methods sensitive to implementation details. Finally, the difficulties of purchasing, building, and installing infrastructure make it challenging for newcomers to contribute to the field.

For robotics research to advance, we clearly need a common way to evaluate and benchmark different algorithms. A good benchmark will not only be fair to all algorithms but also have a low participation barrier: setup to evaluation time should be as low as possible. Efforts like YCB [6] and the Ranking-Based Robotics Benchmark (RB2) [14] have aimed to standardize objects and tasks, but the onus of setting up infrastructure still lies with each lab. A simple way to overcome this is the use of a common physical evaluation site, as the Amazon Picking Challenge [12] and DARPA Robotics Challenges [5, 39, 72] have done. However, the barrier is still high

since participants must set up their own training infrastructure. Both of the above frameworks leave the method development phase unspecified and struggle to provide apples to apples comparisons.

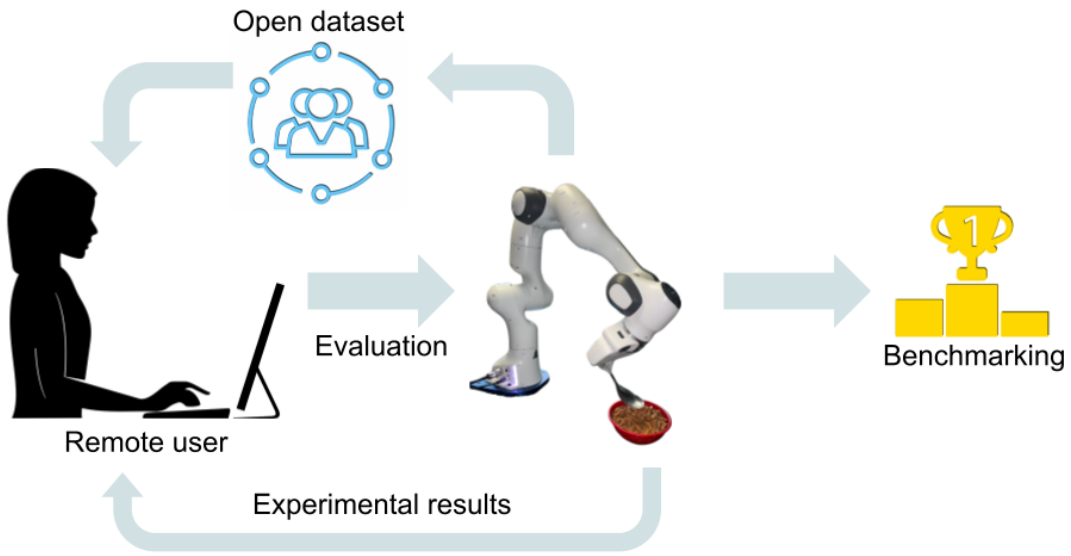


Figure 3.1: **Train Offline, Test Online**: Our benchmark lets remote users test offline learning methods on shared hardware.

Many robot learning algorithms do online training, where a policy is learned concurrently with data collection. One way to standardize online training is with simulation [4, 80, 86, 92]. While simulation mitigates issues with variation across labs, the findings from simulated benchmarks may not transfer to the real world. On the other hand, if we conduct online training in the real world, comparison across labs becomes difficult due to physical differences. In recent years, larger offline datasets have surfaced in robotics [11, 13, 51], and with them the rise of offline training algorithms. From imitation learning to offline reinforcement learning (RL), these algorithms can be trained using the same data and tested in a common physical setup.

Inspired by this observation, we propose a new robotics benchmark called **TOTO (Train Offline, Test Online)**. TOTO has two key components: (a) an offline manipulation dataset to train imitation learning and offline RL algorithms, and (b) a shared hardware setup where users can evaluate their methods now and going forward.

Because all TOTO participants train using the same publicly-released dataset and evaluate on shared hardware, the benchmark provides a fair apples-apples comparison.

TOTO paves a path forward for robot learning by lowering the entry barrier: when designing a new method, a researcher can train their policy on our dataset, evaluate it on our hardware, and directly compare it to the existing baselines for our benchmark. TOTO means no more time devoted to setting up hardware, collecting data, or tuning baselines for one individual's environment. In this paper, we lay out our benchmark design and present the initial methods contributed by benchmark beta testers across the country. These results show that our benchmark suite is challenging yet possible, providing room for growth as users iterate on TOTO.

3.2 Related Work

For a thorough description of work related to remote robotics benchmarking, we refer to the Robotics Cloud concept paper [15]. Here we describe related work specific to our instantiation of a robotics cloud (TOTO).

3.2.1 Shared Tasks and Environments

A necessary step in comparing method performance is evaluation on a common task. Common tasks might mean a standard object set such as YCB [6], which can be distributed to remote labs, allowing for shared metrics like grasp success on these objects. RB2 [14] provides four common manipulation tasks (similar to those we use, described in Section 3.3.2) as well as a framework for comparing and ranking methods across results from multiple labs. Another route is sharing the environment itself, as the Amazon Picking Challenge [12] and DARPA Robotics Challenges [5, 39, 72] have done. Sharing tasks or environments gives metrics by which we can compare approaches. However, users must still develop the approach on their own hardware in their own lab, and recreating identical environment setups is quite challenging.

3.2.2 Shared, Remote Robots

Going one step further, remotely-accessible robots can be shared across the community, enabling method development and evaluation without users acquiring their own

hardware. Georgia Tech's Robotarium [60] allows for remote experimentation of multi-agent methods on a physical robotic swarm, which has been extensively used not just in research but also in education. OffWorld Gym [40] provides remote access to navigation tasks using a mobile robot with closely mirrored simulated and physical instances of the same environment. A recent survey paper [79] provides an overview of robotic grasping and manipulation competitions, including some involving remotely-accessible, shared robots such as [50]. Finally, most closely related to our work, the Real Robot Challenge [21] runs a tri-finger manipulation competition on cube reorientation tasks. The success of the Real Robot Challenge inspires our work, which also allows for evaluation of manipulation tasks on shared robots. Our work, however, is designed to evaluate generalization in robot learning through challenging variations (lighting, unseen test objects, etc.) and an image-based dataset (as opposed to assuming ground-truth state access).

3.2.3 Open-Source Robotics Datasets

Collecting real-world robotics data is challenging and expensive due to physical constraints like environment resets and hardware failures. Thus open-source robotics datasets serve an important role in the field by enabling larger-scale offline robot learning. Some work has improved the way we collect robotics data, such as self-supervised grasping [61] and further parallelization of robots [47]. RoboTurk [51] provides a system for simple teleoperated data collection which can be executed remotely. Much work in robot learning has introduced datasets more generally, such as MIME [74] (8260 demonstrations over 20 tasks), RoboNet [13] (162,000 trajectories collected across 7 robots), and Bridge Data (7,200 demonstrations across 10 environments). However, it is hard to understand the value of these datasets without a common evaluation platform, something that [11] addresses by using simulation to replicate a real-world dataset. In contrast, we address this issue with real-world evaluation that matches the domain of the data collection. Our initial dataset is 2,898 trajectories, but this will grow over time as we add evaluation trajectories collected from users' policies.

3.2.4 Offline Robot Learning

Our benchmark focuses on offline robot learning, including imitation learning and offline RL. Our initial set of baselines is described and contextualized in Section 3.5.2.

3.3 The TOTO Benchmark

Our benchmark focuses on manipulation due to the lack of benchmarking in this area. Our hardware (Section 3.3.1) is set in environments that enable a set of benchmark manipulation tasks described in Section 3.3.2. We collect an initial dataset on these tasks, detailed in Section 3.3.3. Finally, in Section 3.3.4, we present the evaluation protocol for all policies contributed to our benchmark. For more information about our dataset and contributing to the benchmark, please see: https://toto-benchmark.org/.

3.3.1 Hardware

Our hardware includes a Franka Emika Panda robot arm and workstation for real-time inference. A simple joint position control stack runs at 30 Hz. The actions are joint targets, which are converted to motor control signals using a high-frequency PD controller. We also provide an end effector controller in which actions are specified via the position and orientation of the gripper. End effector control using X, Y, Z positions alone is not feasible to solve our tasks: for example, the orientation of the gripper must change as the robot pours. All the results presented in this paper were attained using the joint position controller. We use an Intel D435 RealSense camera for recording RGB-D image observations.

We allow users to opt for a lower control frequency if desired. The training data can be subsampled by taking one of N frames since the actions are in absolute joint angles. We decrease the test time control frequency accordingly.

3.3.2 Tasks

The task suite consists of two manipulation tasks that humans encounter every day, similar to those introduced in prior work [2, 14]. The tasks are pouring and scooping,

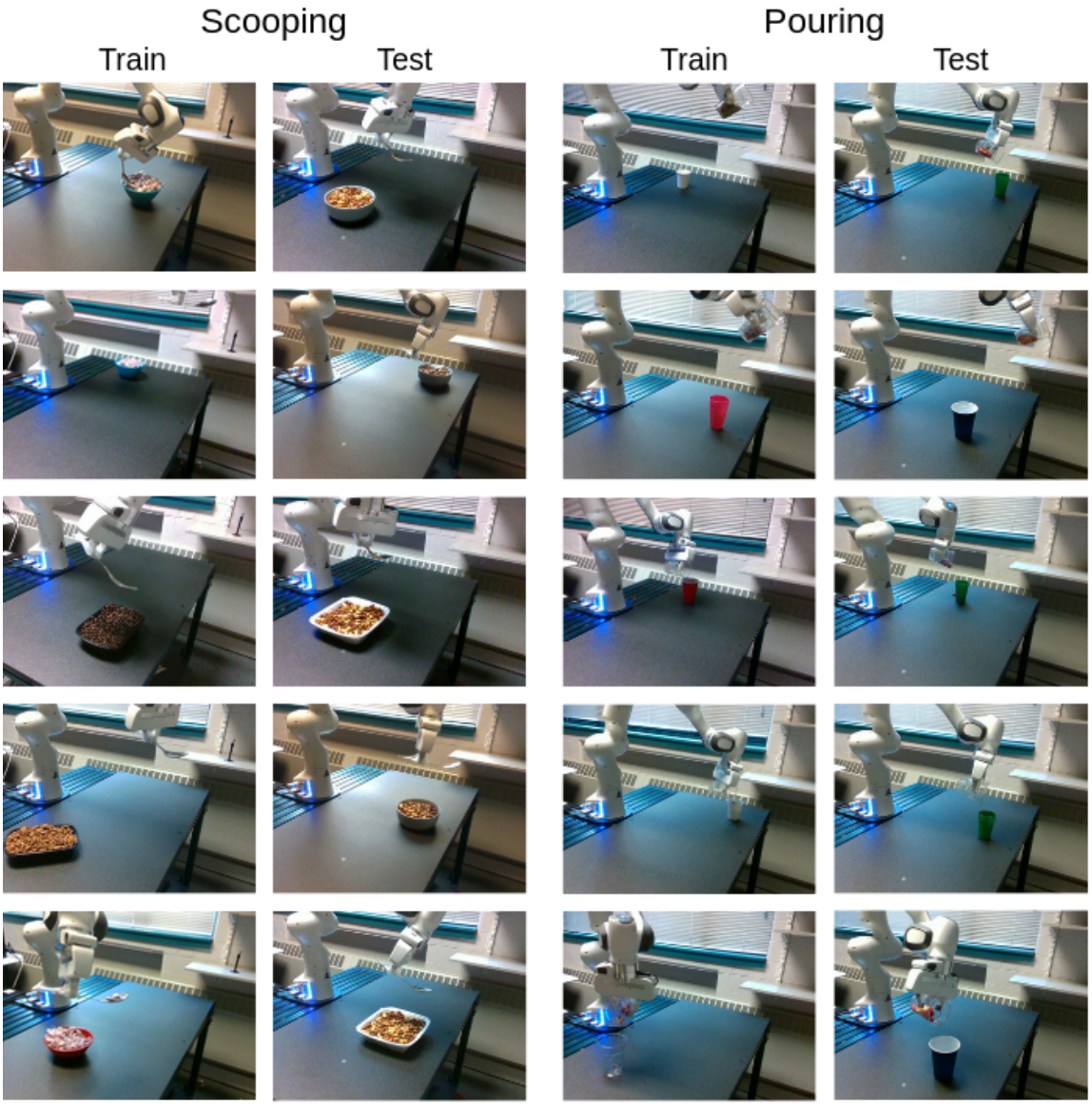


Figure 3.2: **TOTO Task Suite.** Our benchmark tasks are pouring and scooping, similar to those in RB2 [14]. Each involves challenging variations in objects, position, and more.

excluding the easiest and hardest RB2 tasks (zipping and insertion). Example image observations for these tasks are shown in Fig. 3.2. To see the original task designs, please refer to RB2: https://agi-labs.github.io/rb2/. Our tasks differ from those in RB2 in a few ways. We randomize the robot start state at the beginning of each episode. We apply a bit more noise to the target object locations. We use different combinations of objects based on availability. Lastly, we do not normalize the reward: the reward is the weight in grams of the material successfully scooped or poured. For detailed information on the task configurations, such as locations and objects, see our website.

Scooping The robot starts with a spoon in its gripper and a bowl of material on the table. The objective is to scoop material from the bowl into the spoon. The training set includes all combinations of three target bowls, three materials, and six bowl locations (front left, front center, front right, back left, back center, and back right).

Pouring The robot starts with a cup containing granular material in its gripper. The goal is to pour the material into a target cup on the table. The training set includes all combinations of four target cups, two materials, and six target cup locations (same locations as scooping). The cup in the robot gripper is always clear plastic, enabling better perception of the material remaining in the cup.

3.3.3 Dataset

A key pillar of our benchmark is the release of a manipulation dataset. Dataset statistics (number of trials, average trajectory length, success rate, and data collection breakdown) are shown in Table 3.1. We consider a trajectory successful if it obtains a positive reward, and unsuccessful if the reward is zero. The initial release includes between 1000 and 2000 trajectories per task. Pouring data collection using replay and behavior cloning proved challenging to reset (unsuccessful trials require more cleanup), so it was nearly all collected with teloperation. Each trajectory includes images, robot actions (joint angle targets), joint states (joint angles), and rewards. To improve diversity, the data were collected with three techniques, each described below.

Table 3.1: Dataset overview

	Task statistics			Collection technique		
	Trials	Length	Success	Teleop	BC	Replay
`Scooping`	1895	495	0.690	41%	33%	26%
`Pouring`	1003	324	0.977	99%	0%	1%

Teleoperation We collected the majority of trajectories with teleoperation using Puppet [45]. The human controls the robot in an intuitive end effector space using an HTC Vive virtual reality headset and controller. While this teleoperation is theoretically possible to use remotely, we collect the data with the human and robot in the same room, giving the human direct perception of the scene. Our multiple teleoperators have different dominant hands, leading to more diverse data. Most teleoperation trials are successful.

Behavior cloning rollouts After teleoperation trajectories are collected, we train simple, state-based behavior cloning (BC) policies on each target location, so no visual perception is required. We roll out BC trajectories with some noise added to actions at each timestep. The amount of noise varies across trajectories for additional diversity.

Trajectory replay Finally, we also replay individual teleoperated trajectories with added noise. While these might seem overly similar to the original teleoperated trajectories, keep in mind that conditions like lighting also vary with time of day, so this replay still expands the dataset in other ways.

3.3.4 Evaluation Protocol

To evaluate each task, we use two unseen objects (bowls and cups) and one unseen material (mixed nuts for scooping and Starburst candies for pouring). We evaluate three object locations seen during training (front left, front center, front right) and three unseen locations. We evaluate three training seeds of each method. We initialize the robot with a randomly sampled pose at the beginning of each trajectory. However, the robot's initial poses are kept the same across seeds to ensure minimal variance. Combining 2 objects, 1 material, 3 locations, and 3 seeds means that each method

is evaluated across 18 trials each for train and test locations. We report mean and variance of these trials.

3.4 Benchmark Use

Here we introduce the framework for our benchmark. TOTO is designed to make the user workflow (Section 3.4.1) easy for newcomers with well-documented software infrastructure (Section 3.4.2) including examples and tests.

3.4.1 User Workflow

We provide a real-world dataset (Section 3.3.3) collected using our hardware setup (Section 3.3.1). Participants optionally use our software starter kit (Section 3.4.2) and locally train policies of their choosing using this data. Users submit policies through Google Forms for evaluation on our real-world setup. They do not receive the low-level data from these evaluation trials; they simply receive a video showing the policy behavior as well as the reward and success rate.

An engineer supervises the real-world evaluations; thus the evaluation turnaround time is currently around 12 hours (depending on time of day submitted). Our goal is to emphasize offline learning and prevent overfitting, removing the need for real-time results or large quantities of evaluation.

As new users evaluate methods after the paper release, we will post (anonymous) evaluation scores for each attempt on a website leaderboard. We will also periodically add data collected by the users' policies to the original dataset.

3.4.2 Software Infrastructure

Our software starter kit includes documented code and instructions for policy formatting and dataset usage. We have open-sourced baseline code, trajectory data, and pretrained models (see our website). These components ensure that TOTO is easily accessible to a broad portion of the robotics, ML, and even computer vision communities.

We adapt the agent format from [32], which requires a `predict` function taking

in the observation and returning the action. We use a standard config format and require an `init_agent_from_config` function to create the agent.

We provide users with code for training an example image-based BC agent and a docker environment which wraps the minimum required dependencies to run this code. Users can optionally extend the docker containers with additional dependencies. We also provide a stub environment for users to locally verify whether their agent's predictions are compatible with our robot environment. This setup allows users to resolve all agent format and library dependency issues before submitting agents for evaluation.

3.5 Baselines

We highlight the importance of establishing a benchmark by running two sets of experiments: (a) what is a good visual representation for manipulation? and (b) what is a good offline algorithm for policy learning? To test the benchmark infrastructure, we have solicited baseline implementations for both experiments from several labs.

3.5.1 Visual Representation Baselines

A core unanswered question, due to the lack of benchmarking, is what is a good visual representation for manipulation? Is ResNet trained on ImageNet great or do self-supervised approaches outperform supervised models? We evaluate five visual representations provided by TOTO users from multiple labs. Two are trained on our data (in-domain) and three are generically pretrained.

`BYOL` (Bootstrap Your Own Latent) [23] is a self-supervised representation learning method trained on our dataset. The BYOL representation embedding size is 512.

`MoCo (Generic)` refers to the Momentum Contrast (MoCo) model trained on ImageNet [27], while `MoCo (In-Domain)` is trained on our data with crop-only augmentations [59].

`Resnet50` refers to the model trained with supervised learning on ImageNet [26].

`R3M` (Reusable Representations for Robot Manipulation) [56] is trained on Ego4D [22] with time-contrastive learning and video-language alignment. For `R3M`, `MoCo`, and `Resnet50`, we use the 2048-dimensional embedding vector following the fifth

convolutional layer.

These representations performed the best among a larger set of vision models on which we ran an initial brief analysis (including offline visualizations and BC rollouts). Additional representations that performed less well (and therefore are not included as baselines) included CLIP [65] and a lower-level MoCo model (from the third layer instead of the fifth).

3.5.2 Policy Learning Baselines

Remote users have contributed the below policy learning baselines, which span the spectrum from nearest neighbor querying to BC to offline RL. They were selected according to each TOTO contributor's expertise with approach coverage in mind. All methods use RGB image observations, and some run these images through a frozen, pretrained vision model before passing the resulting embedding to a policy.

`BC` is trained on top of each vision representation baseline. Closed-loop BC predicts a new action every timestep, while open-loop BC predicts a sequence of actions to execute without re-planning. Our BC baseline is *quasi* open-loop: training trajectories are split into 50-step action sequences, and the policy is trained to predict such a sequence given the current observation. During evaluation, these 50 actions are executed between each prediction step. We find that this performs better than closed-loop or open-loop alone: closed-loop struggles without history, and open-loop is challenging with our variable-length tasks. We filter the training data to only include trajectories with nonzero reward [10].

`VINN` (Visual Imitation through Nearest Neighbors) [58] is a nearest neighbor policy using an image encoder trained with `BYOL` [23]. While using nearest neighbors as a policy has been previously explored [53], this approach alone does not scale well to high-dimensional observations like images. BYOL maps the high-dimensional observation space to a low dimension to obtain a robust policy. VINN was originally closed-loop (query and execute a new action at each timestep), but in this work we mirror the 50-step quasi open-loop approach used in the BC baseline (described above).

`IQL` (Implicit Q-learning) [37] uses the open-source implementation from the `d3rlpy` package [73]. We use `MoCo (In-Domain)` as a frozen visual representation

since it performed the best in our comparison of representations with BC. We concatenate the frozen image embeddings with the robot's joint angles as the input state to the model.

`DT` (Decision Transformers) [10] recasts offline RL as a conditional sequence modeling task using transformers. Similar to BC, it is trained to predict the action in the dataset, but conditions on the trajectory history as well as target return (desired level of performance). We use the Hugging Face DT implementation. The model receives an RGB image and the robot's joint angles: the former is embedded using `MoCo (In-Domain)` and concatenated with the latter at each step. DT uses a sub-sampling period of 8 and a history window of 10 frames. For inference and evaluation, the target return prompt is approximately chosen as the mean return from the top 10% of trajectories in the dataset for each task.

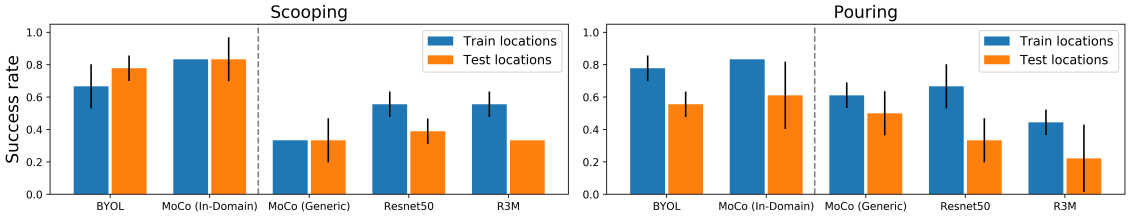


Figure 3.3: **Vision representation comparison with BC.** Models trained on our data (left of dashed line) perform better than generic ones (right of dashed line), and results tend to be better for training object locations than unseen test locations.

3.6 Experimental Results

3.6.1 Visual Representation Comparison Using BC

Our first set of experiments compares BC agents using the vision representations detailed in Section 3.5.1 and evaluated with the protocol described in Section 3.3.4. The success rates across all representations and tasks are visualized in Fig. 3.3, and the numerical rewards are presented in Table 3.2.

These results show that finetuning the MoCo model on our data outperforms the generic version, as expected. `MoCo (In-Domain)` achieves the highest success rate and average reward on both tasks, followed by `BYOL`, the other in-domain model. In

general, the relative performance between models is mostly consistent across tasks. Resnet50 and `MoCo (Generic)` perform slightly better on pouring than on scooping.

Fig. 3.3 also visualizes performance differences due to object locations. Locations seen during training perform better, but performance does not degrade significantly, suggesting that the representations have a generalizable notion of where the target object is. Surprisingly, the two representations trained on our data (`MoCo (In-Domain)` and `BYOL`) perform equally good or even slightly better on unseen locations for scooping.

Table 3.2: Performance of vision representations with BC across train and test locations.

	Method	Scooping		Pouring	
		Reward	Success %	Reward	Success %
In Domain	`BYOL`	4.39	72.2%	20.22	66.6%
	`MoCo`	**7.42**	**83.3%**	**22.86**	**72.2%**
Out of Domain	`MoCo`	2.11	33.3%	14.89	55.5%
	`ResNet50`	2.83	47.2%	18.86	50.0%
	`R3M`	2.97	44.4%	6.94	33.3%

3.6.2 Policy Learning Results

Table 3.3 shows the comparison of policy learning methods (described in 3.5.2) evaluated on TOTO. Due to compute constraints, we have 1 and 2 seeds for `DT` and `IQL` respectively. We compensate by duplicating the evaluation of these seeds to keep the number of trials consistent. The results are visualized in Fig. 3.4. We find that `VINN` performs the best in train locations. We also note that offline-RL approaches (especially IQL) achieve some success unlike in RB2[14]. This is likely due to a larger and more diverse dataset than RB2, which contributes to better offline RL performance.

In experiments, we found that scooping proves challenging due to a non-markovian aspect of the task: the spoon is above the bowl both before and after scooping. Thus we would expect open-loop methods (`BC`, `VINN`) and those with history (`DT`) to perform better than others in this setting. While `BC` and `VINN` achieve competitive performance

on scooping, DT only achieves moderate success on scooping and does not see any positive rewards on pouring. Meanwhile, IQL provides decent performance without history on a non-markovian task.

Comparing the train and test location results for policy learning proves interesting. VINN performs the best on train locations, but it struggles on unseen locations, since it selects actions using the nearest neighbor trajectory from the training data. All other methods also experience some level of degradation when moving to unseen locations, leaving one clear direction for method improvement using TOTO.

Table 3.3: TOTO policy learning results across train and test locations.

Method	Scooping		Pouring	
	Reward	Success %	Reward	Success %
BC + MoCo	7.42	**83.3%**	**22.86**	**72.2%**
VINN	**7.89**	63.9%	21.75	47.2%
IQL	6.08	47.2%	9.86	38.9%
DT	2.83	27.8%	0.00	0.0%

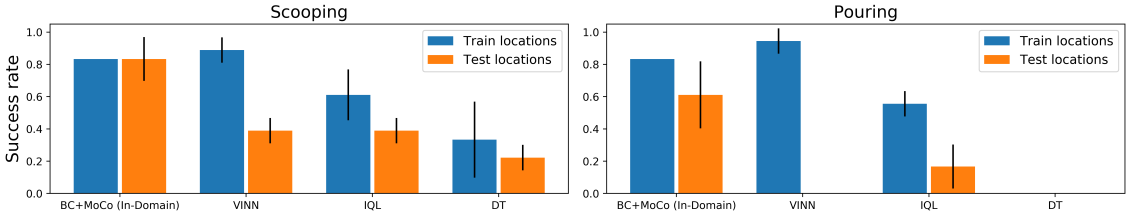


Figure 3.4: **Evaluating offline policy learning results.** VINN sees the best performance on train locations, but its performance degrades on unseen locations, as does the performance of other methods.

3.6.3 Dataset Size Ablation

To understand the impact of dataset size on policy learning performance, we perform an ablation in which we train BC on the scooping task with varying amounts of data. We sort the scooping trajectories by reward and train policies with the top 5%, 25%, 50%, 100% of the data, as well as all successful trajectories with positive rewards (∼70%). This sorting by reward ensures that policies trained in the small-data regime

are not overcome by unsuccessful trajectories. We present the dataset size ablation results in Table 3.4.

Table 3.4: Dataset size ablation with BC on scooping.

Dataset size	Reward	Success Rate
5%	2.89	38.9%
25%	5.94	72.2%
50%	6.22	77.8%
Successes (∼70%)	8.06	83.3%
100%	5.00	72.2%

The *Successes* number uses the same policy as the BC policy in Table 3.3, but we evaluate it again with the ablations to ensure minimal variance in conditions. As expected, training on more data generally leads to a higher success rate. Training on all of the data (including unsuccessful trajectories) leads to a lower reward than training on only the successful trajectories, also unsurprising given the use of BC to learn the policies in this ablation (we might expect offline RL to improve with the inclusion of unsuccessful trials).

Overall, these ablation results suggest that the TOTO dataset size is the right order of magnitude in terms of policy learning. We have reached the point of diminishing returns: training on 50% versus 70% of the data does not substantially improve performance. However, additional data might still improve visual representation learning.

3.6.4 Metrics for Offline Policy Evaluation

A TOTO user might wish to sanity check their policy before submitting it for real-world evaluation or otherwise have performance metrics to guide offline tuning. Here we present simple example metrics for offline evaluation: action similarity to a validation set of expert demonstrations using both joint angle error and end effector pose error. From a chosen validation set of 100 trajectories, we estimate the joint angle error and end effector error by computing the mean squared error between agent's predicted actions and actual actions for all samples.

Fig. 3.5 shows these validation metrics on BC checkpoints throughout training and the real-world reward evaluated on four representative checkpoints. The reward

increases as the validation error metrics decrease, matching expectations. These metrics capture overfitting: the overtrained policy from 2,000 epochs shows a significant decrease in real-world reward and likewise has higher validation error. While offline metrics alone should not fully guide the development of an algorithm, it provides a signal as to whether the policy might achieve any success in the real world.

Figure 3.5: **Comparing offline evaluation to online performance.** While offline evaluation is imperfect, it provides a sanity check to the user, guiding development at a higher frequency than real-world evaluation.

3.7 Discussion

The main goal of this work is to introduce TOTO, our robotics benchmark. We presented a broad initial set of baselines containing both vision representations and policy learning approaches, which can be built off of by future TOTO users. Notably, these baselines were contributed in the same way that TOTO will be used in the future: by collaborators who locally train policies and submit them for remote evaluation on shared hardware. This shows the feasibility of our user workflow. The initial baseline results show the challenging nature of our tasks, especially with respect to generalization. By using TOTO as a community, we can more quickly iterate on ideas and make progress on the real-world bottlenecks to robot learning.

3.7.1 Limitations and Future Work

The evaluation protocol currently has manual steps: we measure the material transferred during pouring and scooping to compute rewards and reset by returning the material to the original object. We do see future potential to automate reward measurements and resets, such as by adding a scale beneath the target object and using an additional robot to reset the transferred materials. Spills of the transferred material, however, might still require manual intervention.

We plan to expand the evaluation setup to include additional robots. This would help us meet the increasing demand in evaluations as more users adopt the benchmark. One challenge will be visual differences across robots, but we plan to collect additional demonstrations on new robots, and this would be an opportunity to expand the set of tasks as well (we could designate one robot per task).

As user demand further grows, we will implement an evaluation job queue which prioritizes evaluation requests from different users and schedules the jobs based on the number of robots currently available.

Chapter 4

Conclusions

In conclusion, I have presented two methods tackling the challenges for establishing benchmarks on physical robots. Both methods have shown effectiveness in evaluating robot learning algorithms' generalization capabilities. While Real-ORL focuses more on the generalization and transfer abilities of offline reinforcement learning algorithms, TOTO opens a portal for the entire community to evaluate the generalization capabilities of visual and policy learning methods in a fair and direct way.

In the first work, Real-ORL, we conducted an empirical study of representative ORL algorithms on real-world robotic learning tasks. The study encompassed three representative ORL algorithms (along with behavior cloning), four table-top manipulation tasks with a Franka-Panda robot arm, 3000+ train trajectories, 3500+ evaluation trajectories, and 270+ human labor hours. Through our extensive ablation studies, we find that (1) even for in-domain tasks with abundant amount of high-quality data, IQL can be a competitive baseline against the best behavior cloning policy, (2) for out-domain tasks, ORL algorithms were able to generalize to task regions with low data-support and to dynamic tasks, (3) the performance changes of ORL after leveraging heterogeneous data are likely to vary by agents, the design of the task, and the characteristics of the data, (4) certain heterogeneous task-agnostic data could provide overlapping data support and enable transfer learning, allowing ORL agents to improve their own performance and, in some cases, even surpass the best in-domain agents. Overall, (5) the best agent for each task is either an ORL algorithm or a tie between ORL and BC. Our rigorous evaluations indicate that

even in out-of-domain multi-task data regime, (more realistic in real world setting) offline RL is an effective paradigm to leverage out of domain data.

In the second work, TOTO, we presented a broad initial set of baselines containing both vision representations and policy learning approaches, which can be built off of by future TOTO users. Notably, these baselines were contributed in the same way that TOTO will be used in the future: by collaborators who locally train policies and submit them for remote evaluation on shared hardware. This shows the feasibility of our user workflow. The initial baseline results show the challenging nature of our tasks, especially with respect to generalization. By using TOTO as a community, we can more quickly iterate on ideas and make progress on the real-world bottlenecks to robot learning.

Appendix A

Appendix for Real-ORL

A.1 Canonical Task Setup

We considered four canonical tasks: `reach`, `slide`, `lift` and `PnP` (pick-and-place). To apply ORL, each task can be formulated as an MDP. The state contains the joint position of the robot, the gripper open position ($\mathcal{R} \sim [0, 0.08]$), (optionally) the velocity of the joints, (optionally) the tracked tag position and a goal position. To facilitate RL training, we came up with a continuous reward function for each task $r : state \to \mathcal{R}$, as shown in Table A.1, considering the position of the gripper x, the position of tracked AprilTag t (if exists), the position of goal g, the Euclidean distance function dis between two 3D coordinates, a convenient function $height$ to denote the height of a given coordinates. While the reward for `reach` and `slide` are naturally smaller than 1, we explicitly cap the maximum reward for `lift` to be 1 since we don't encourage agents to lift up the lid arbitrarily high. We don't cap the `PnP` reward since we encourage the `PnP` policy to be distinguished from the `slide` policy with a height bonus $height(t)$.

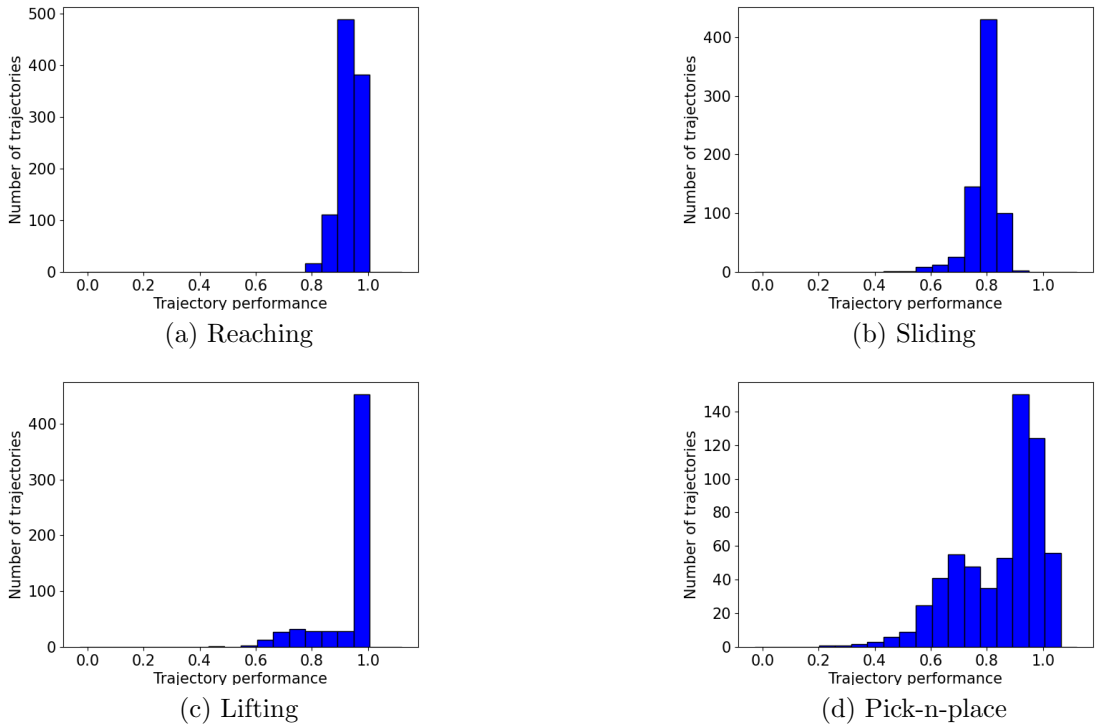
We used heuristic policies to collect the demonstration data, as described in Sec. 2.3. Our policies have a reasonable success rate *accomplishing* the task but is not designed to be optimal in solving the MDP. To evaluate and compare between agents, we instead report the maximum reward over the trajectory as a proxy of the task completion ("score"). We report our heuristic policies' accumulated reward average over trajectories and the score.

Table A.1: Characteristics of task and collected data.

Task	$r(s)$	$\sum r(s)$	Score
Reach	$1 - dis(g - x)$	173	0.99
Slide	$1 - (2 * dis(g - t) + dis(t - x))$	223	0.93
Lift	$min(1, 0.57 - dis(t - x) + height(t))$	167	1
Pick-n-place	$1 - (dis(g - t) + 2 * dis(t - x)) * 0.9 + height(t)$	281	1.09

A.2 Dataset

We also attach our collected data's score distribution on each task to demonstrate our dataset's overall quality. From Figure A.1, we can see that the score distribution for each task skew heavily to the left, which means that most trajectories in our dataset are near-optimal and are suitable for imitation learning. In Appendix. A.5 we further verify this.



(a) Reaching (b) Sliding

(c) Lifting (d) Pick-n-place

Figure A.1: Score distribution for each task of our dataset.

A.3 Open Source Code and Dataset

We open source our code and publish the dataset at our website.

A.4 Training Details

Our code base was built upon the author's implementation of MOReL [33] and the D3RLPY [73] library. We used the same fixed random seed for all our experiments (123), unless otherwise specified. For hyperparameter tuning, we always started training using the default hyperparameters. If the training loss reported by the agent did not converge, we adjusted the learning rate and retrain, up to 5 agents, till we find a model that converge or have been trained for $5,000,000$ steps using batch size 2048. For model whose training loss exploded (e.g., AWAC), we choose an checkpoint from earlier of the training when the loss were relatively stable for $100,000$ steps (frequently, this was an agent that finished about half a million to a million training steps). Surprisingly, when evaluated on real robot, models that reported convergence did not necessarily perform better than model that did not converge.

Practicality of Training and Tuning BC was the cheapest to train ($\sim$ 3min) and easiest to converge (no additional tuning required). MOReL was the second shortest to train ($\sim$ 4 hours); most MOReL agents were able to converge, judged by the reward of trajectories generated by the learned dynamics model. AWAC agents took longer to train ($\sim$ 12 hours) and had the most trouble converging (8 of the 16 agents in the ablation table could not converge in allocated trials). IQL agents took the longest to train (10 $\sim$ 24 hours) but had more success converging. Though loss convergence during training or a good reward estimated by the learned dynamics model or learned value function cannot indicate the agent's true performance, it is helpful for selecting an agent to test. Since some AWAC agents had trouble converging, we selected an earlier checkpoint before loss explosion and documented their performance, which, surprisingly, yielded higher reward than some agents that reported convergence. We leave it to future work to investigate this phenomenon.

A.5 Training Behavior Cloning with Top-K% Trajectories

To ensure that our dataset contains high quality trajectories that is sufficient to train behavior cloning, we launched new experiments training behavior cloning using only the Top-k % of the best trajectories. In Figure. A.1, we plot the distribution of performance of our data for each task. For `reach`, `slide`, `lift`, 90% of trajectories complete the task with good scores (¿ 0.75). For `PnP` (our most difficult task), 50% of our collected trajectories completed the task (scores ¿ 0.8).

Thus we train BC for `reach`, `slide`, `lift` on Top-90% of data and train BC for `PnP` on Top-50%,70%,90% of data and observe that, BC in our experiments benefit from using the full dataset.

Table A.2: Performance of BC agents trained with Top-k% data.

Task	**Top-k%**	**#Trajs**	**Threshold for Demo**	**Score**	**Original Score** (BC with full data)
`reach`	90	900	0.909	0.899 ± 0.037	0.924 ± 0.048
`slide`	90	657	0.774	0.659 ± 0.152	0.681 ± 0.147
`lift`	90	554	0.787	0.784 ± 0.157	0.823 ± 0.177
`PnP`	50	304	0.935	0.723 ± 0.217	0.818 ± 0.185
	70	426	0.792	0.789 ± 0.290	0.818 ± 0.185
	90	548	0.656	0.789 ± 0.204	0.818 ± 0.185

A.6 Sweeping of Random Seeds

We evaluated an addition of 28 agents for 340 trajectories for a total of 70 hours including training and testing to inspect how the scores for critical agents (i.e., the best agents for a category) would vary by random seeds. We now have 3 seeds for each of the following agents:

1. The Best Agents for each task in Table 2.2
2. The Second Best Agents for each task in Table 2.2

3. ORL agents with out-domain datasets in in Table 2.5

The original agents are trained with seed 123, we trained the additional agents with seed 122 and seed 124. Each seed is evaluated on 12 trajectories. The results are listed and we observe that ∼60% of newly trained agents change score by less than 1%, ∼90% of agents change by less than 2%, and the maximum change was 6% from one agent (whose score change does not affect our conclusion).

Table A.3: Seed sweeping results of the best agents for each task in Table 2.2.

Best Agents in Table 2.2	**Seed 122**	**Seed 124**	**Seed 123** (original seed)	**Means w/ 3 seeds**	**Mean diff**
AWAC, DeltaVel,`reach`	0.920 ± 0.031	0.919 ± 0.066	0.935 ± 0.032	0.925 ± 0.047	0.01 (1.07%)
IQL, DeltaVel, `slide`	0.781 ± 0.038	0.763 ± 0.044	0.757 ± 0.095	0.767 ± 0.065	-0.01 (-1.32%)
IQL, DeltaVel, `lift`	0.877 ± 0.166	0.878 ± 0.158	0.884 ± 0.120	0.880 ± 0.149	0.004 (0.45%)
BC, AbsVel, `PnP`	0.819 ± 0.199	0.800 ± 0.195	0.836 ± 0.157	0.818 ± 0.185	0.018 (2.15%)

Table A.4: Seed sweeping results of the second-best agents for each task in Table 2.2.

Second Best in Table 2.2	**Seed 122**	**Seed 124**	**Seed 123** (original seed)	**Means w/ 3 seeds**	**Mean diff**
MOREL, DeltaVel, `reach`	0.919 ± 0.034	0.908 ± 0.042	0.925 ± 0.028	0.917 ± 0.036	0.008 (0.86%)
BC, DeltaVel, `reach`	0.921 ± 0.051	0.917 ± 0.055	0.934 ± 0.032	0.924 ± 0.048	0.01 (1.07%)
BC, AbsVel, `slide`	0.699 ± 0.125	0.698 ± 0.120	0.645 ± 0.18	0.681 ± 0.147	-0.036 (-5.58%)
MOREL, DeltaVel, `slide`	0.655 ± 0.157	0.602 ± 0.180	0.629 ± 0.136	0.629 ± 0.160	0 (0%)
AWAC, DeltaVel,`slide`	0.757 ± 0.068	0.703 ± 0.108	0.739 ± 0.144	0.732 ± 0.113	0.007 (0.95%)
BC, AbsVel, `lift`	0.821 ± 0.192	0.832 ± 0.177	0.818 ± 0.161	0.823 ± 0.177	-0.005 (0.61%)

Table A.5: Seed sweeping results of ORL agents with out-domain datasets in Table 2.5.

ORL in Table 2.5	**Seed 122**	**Seed 124**	**Seed 123** (original seed)	**Means w/ 3 seeds**	**Mean diff**
AWAC on `PnP` w/ slide+lift	(diverged)	0.811 ± 0.103	0.815 ± 0.134	0.813 ± 0.121	0.002 (0.25%)
AWAC on `PnP` w/ slide+lift+pnp	0.759 ± 0.180	0.773 ± 0.204	0.742 ± 0.175	0.758 ± 0.188	-0.016 (-2.16%)
IQL on `PnP` w/ slide+lift	0.838 ± 0.103	0.847 ± 0.117	0.843 ± 0.120	0.842 ± 0.114	0.001 (0.12%)
IQL on `PnP` w/ slide+lift+pnp	0.842 ± 0.170	0.826 ± 0.211	0.829 ± 0.163	0.833 ± 0.183	-0.004 (-0.48%)
MOREL on `lift` w/ slide+lift+pnp	0.879 ± 0.124	0.904 ± 0.119	0.906 ± 0.151	0.896 ± 0.133	-0.01 (-1.1%)

A.7 Statistical Significance of Conclusions

In this section we report the statistical significance of the conclusions we drew from our empirical study. To evaluate every trained agent for every task, we collected at least 12 trajectories and calculated their scores. For best agents or agents used in comparison, we trained them with additional seed to collect a total of 36 trajectories.

We note that the distribution of scores is unknown. We cannot exclude the possibility of the distribution being skewed, as the agent could perform better in a certain task region because of the nature of the task. Therefore, we conducted the Wilcoxon signed T-test for paired samples to calculate the p-value.

With $p < 0.1$, we reject the null hypothesis that the two models' have identical scores. Tasks and application-domains determine the confidence level requirements for any application. This often requires domain knowledge and might not transfer between different applications even for the same task. For openness and interpretability, we clearly outline our statistical tests and list our p-values, leaving it up to the readers to justify their statistical significance required for their applications. We found that:

1. On in-domain tasks, we observe that: On the simplest task `reach`, all agents achieved comparable performance ($p > 0.1$). On `slide`, IQL outperformed BC ($0.77 > 0.68, p = 0.001$). On `lift`, IQL outperformed BC ($0.88 > 0.82, p = 0.041$). On `PnP`, BC outperformed the best ORL agent ($p = 0.016$).

2. Testing agent's ability to generalize to task space lacking data support, we verify that MOREL and AWAC achieved comparable or better performance than BC for regions lacking data support (MOREL: $0.79 \sim 0.76, p = 0.50$, AWAC: $0.83 > 0.76, p = 0.006, p = 0.25$), despite that these ORL agents were having poorer performance on regions that have more data support ($0.67 < 0.78, p = 0.062$).
3. In terms of leveraging multi-task data, MOREL has clearly benefited from inclusion of more data. On `slide`, the model achieved significantly higher performance when using combined data from three tasks ($0.63 \sim 0.72, p = 0.027$). On `lift`, the model achieved significantly higher performance when using combined data from three tasks ($0.68 \rightarrow 0.90, p = 0.003$). For AWAC, it gained performance on `lift`($0.86 \rightarrow 0.90, p = 0.059$). IQL had most success for `PnP` task after leveraging `slide`+`lift` data ($p = 0.001$).

Appendix B

Appendix for TOTO

B.1 Task Specifications

Our benchmark contains two manipulation tasks: scooping and pouring. Fig. B.1 shows the train and test objects for each task. The containers have varied sizes, shapes, materials, and colors. The materials being scooped or poured have diverse appearances, granularities, and densities. This variation enables us to evaluate the generalization capabilities of both visual representations and learned policies.

Fig. B.2 shows the train and test locations for the center of the container in both tasks. These locations are distributed across a workspace measuring $60cm$ x $110cm$. To ensure evaluation consistency, we have marked each location for future reference.

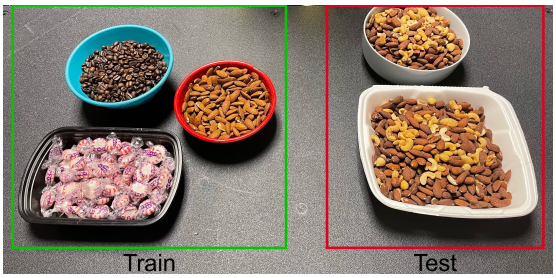
B.2 Dataset

We compute the reward distribution of our dataset for both tasks (Fig. B.3). In the pouring task, the presence of multiple peaks in the distribution is attributed to the weight differences of the materials poured during training.

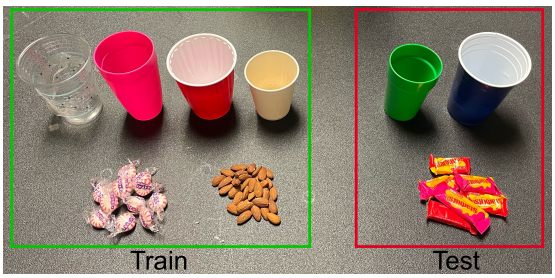
We note that the reward values depicted in Fig. B.3 are not normalized. Instead, they represent the weight in grams of the material successfully scooped or poured. This unnormalized representation provides a direct measure of the task performance in terms of the weight of the manipulated material.

(a) Scooping Task

(b) Pouring Task



(c) Scooping Objects



(d) Pouring Objects

Figure B.1: **Benchmark Suite.** Our benchmark includes two tasks: scooping (left) and pouring (right). The bottom images display the train and test objects for each task, respectively.

B.3 Get Started with TOTO

We are committed to maintaining the TOTO setup in the long term and continuously seeking additional baselines. We warmly invite the academic and research community to participate in the TOTO benchmark challenges, which include:

- **Visual Representation Model Challenge**: In this challenge, participants are encouraged to train and submit Behavior Cloning (BC) agents that utilize a pre-trained visual representation model.
- **Agent Policy Challenge**: Participants can choose to either develop a custom visual representation model or utilize the visual representations provided by TOTO. The challenge focuses on designing and submitting agent policies that

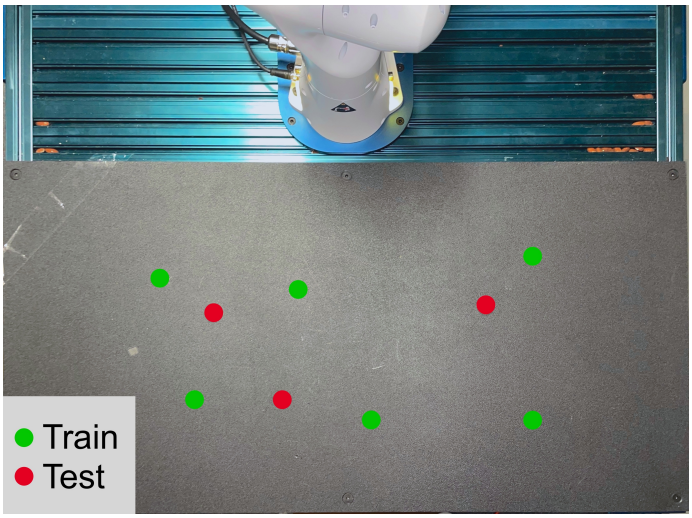


Figure B.2: **Train and Test Locations.** The green locations are used in the training data. The red locations are only used at test time.

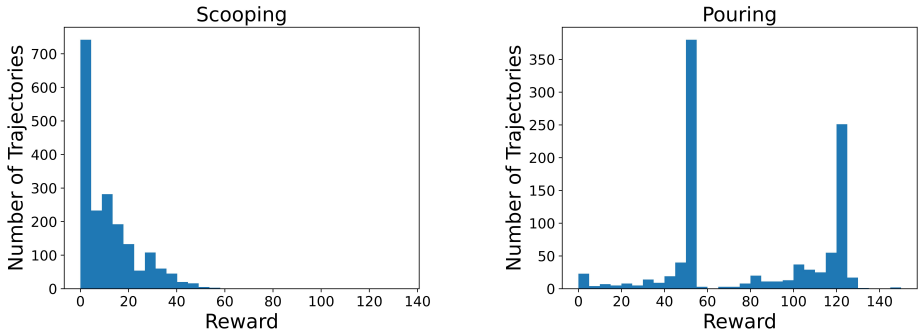


Figure B.3: **Reward Distribution.** Each histogram depicts the task rewards in the training dataset.

demonstrate effective manipulation skills.

For comprehensive instructions on how to get started, as well as to access our code and dataset, we encourage you to visit our website at https://toto-benchmark.org/.

Bibliography

[1] Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning. *ArXiv*, abs/2008.05556, 2021. 2.2

[2] Shikhar Bahl, Abhinav Gupta, and Deepak Pathak. Hierarchical neural dynamic policies. *arXiv preprint arXiv:2107.05627*, 2021. 3.3.2

[3] Mohak Bhardwaj, Balakumar Sundaralingam, Arsalan Mousavian, Nathan D. Ratliff, Dieter Fox, Fabio Ramos, and Byron Boots. STORM: An integrated framework for fast joint-space model-predictive control for reactive manipulation. 2021. 2.3

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016. 3.1

[5] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The DARPA urban challenge: autonomous vehicles in city traffic*, volume 56. Springer, 2009. 3.1, 3.2.1

[6] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *International Conference on Advanced Robotics*, pages 510–517. IEEE, 2015. 3.1, 3.2.1

[7] Jonathan Chang, Masatoshi Uehara, Dhruv Sreenivas, Rahul Kidambi, and Wen Sun. Mitigating covariate shift in imitation learning via offline data without great coverage. *ArXiv*, abs/2106.03207, 2021. 2.2

[8] Yevgen Chebotar, Karol Hausman, Yao Lu, Ted Xiao, Dmitry Kalashnikov, Jacob Varley, Alex Irpan, Benjamin Eysenbach, Ryan C. Julian, Chelsea Finn, and Sergey Levine. Actionable models: Unsupervised offline reinforcement learning of robotic skills. *ArXiv*, abs/2104.07749, 2021. 2.2

[9] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision Transformer: Reinforcement Learning via Sequence Modeling. 2021. 2.1, 2.2, 2.7

[10] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha

Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in Neural Information Processing Systems*, 34:15084–15097, 2021. 3.5.2

[11] Jack Collins, Jessie McVicar, David Wedlock, Ross Brown, David Howard, and Jürgen Leitner. Benchmarking simulated robotic manipulation through a real world dataset. *IEEE Robotics and Automation Letters*, 5(1):250–257, 2019. 3.1, 3.2.3

[12] Nikolaus Correll, Kostas E Bekris, Dmitry Berenson, Oliver Brock, Albert Causo, Kris Hauser, Kei Okada, Alberto Rodriguez, Joseph M Romano, and Peter R Wurman. Analysis and observations from the first amazon picking challenge. *IEEE Transactions on Automation Science and Engineering*, 15(1):172–188, 2016. 3.1, 3.2.1

[13] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2019. 3.1, 3.2.3

[14] Sudeep Dasari, Jianren Wang, Joyce Hong, Shikhar Bahl, Yixin Lin, Austin S Wang, Abitha Thankaraj, Karanbir Singh Chahal, Berk Calli, Saurabh Gupta, et al. Rb2: Robotic manipulation benchmarking with a twist. In *NeurIPS Datasets and Benchmarks Track*, 2021. (document), 3.1, 3.2.1, 3.3.2, 3.2, 3.6.2

[15] Victoria Dean, Yonadav G Shavit, and Abhinav Gupta. Robots on demand: A democratized robotics research cloud. In *Conference on Robot Learning*, pages 1769–1775. PMLR, 2022. 3.2

[16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009. 3.1

[17] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022. 2.3

[18] Justin Fu, Aviral Kumar, Ofir Nachum, G. Tucker, and S. Levine. D4RL: Datasets for Deep Data-Driven Reinforcement Learning. *ArXiv*, abs/2004.07219, 2020. 2.1, 2.1, 2.2, 2.3

[19] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. 2.2, 2.4, 2.7

[20] Scott Fujimoto, David Meger, and Doina Precup. Off-Policy Deep Reinforcement Learning without Exploration. *CoRR*, abs/1812.02900, 2018. 2.1, 2.2

[21] Niklas Funk, Charles Schaff, Rishabh Madan, Takuma Yoneda, Julen Urain De Jesus, Joe Watson, Ethan K Gordon, Felix Widmaier, Stefan Bauer, Siddhartha S Srinivasa, et al. Benchmarking structured policies and policy optimization for real-world dexterous object manipulation. *arXiv preprint arXiv:2105.02087*, 2021. 3.2.2

[22] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Conference on Computer Vision and Pattern Recognition*, pages 18995–19012. IEEE, 2022. 3.5.1

[23] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33:21271–21284, 2020. 3.5.1, 3.5.2

[24] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017. 2.3

[25] Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Tom Le Paine, Sergio Gómez Colmenarejo, Konrad Zolna, Rishabh Agarwal, Josh Merel, Daniel Mankowitz, Cosmin Paduraru, et al. Rl unplugged: Benchmarks for offline reinforcement learning. *arXiv preprint arXiv:2006.13888*, 2020. 2.1, 2.1, 2.2

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 770–778. IEEE, 2016. 3.5.1

[27] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Conference on Computer Vision and Pattern Recognition*, pages 9729–9738. IEEE, 2020. 3.5.1

[28] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *NeurIPS*, 2019. 2.1

[29] Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Àgata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind W. Picard. Way Off-Policy Batch Deep Reinforcement Learning of Implicit Human Preferences in Dialog. *CoRR*, abs/1907.00456, 2019. 2.2

[30] Firas Jarboui and Vianney Perchet. Offline inverse reinforcement learning. *arXiv preprint arXiv:2106.05068*, 2021. 2.2

[31] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico

Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *ArXiv*, abs/2104.08212, 2021. 2.2

[32] Liyiming Ke, Jingqiang Wang, Tapomayukh Bhattacharjee, Byron Boots, and Siddhartha Srinivasa. Grasping with chopsticks: Combating covariate shift in model-free imitation learning for fine manipulation. In *International Conference on Robotics and Automation*. IEEE, 2021. 2.2, 3.4.2

[33] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. MOReL : Model-Based Offline Reinforcement Learning. In *NeurIPS*, 2020. 2.1, 2.2, 2.4, A.4

[34] Ksenia Konyushkova, Konrad Zolna, Yusuf Aytar, Alexander Novikov, Scott Reed, Serkan Cabi, and Nando de Freitas. Semi-supervised reward learning for offline reinforcement learning. *arXiv preprint arXiv:2012.06899*, 2020. 2.2

[35] Ksenia Konyushova, Yutian Chen, Thomas Paine, Caglar Gulcehre, Cosmin Paduraru, Daniel J Mankowitz, Misha Denil, and Nando de Freitas. Active offline policy selection. *Advances in Neural Information Processing Systems*, 34: 24631–24644, 2021. 2.1

[36] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021. 2.1, 2.2, 2.2, 2.4, 2.5.2

[37] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021. 3.5.2

[38] Oliver Kroemer, Scott Niekum, and George Dimitri Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *Journal of machine learning research*, 22(30), 2021. 2.3

[39] Eric Krotkov, Douglas Hackett, Larry Jackel, Michael Perschbacher, James Pippine, Jesse Strauss, Gill Pratt, and Christopher Orlowski. The darpa robotics challenge finals: Results and perspectives. *Journal of Field Robotics*, 34(2): 229–240, 2017. 3.1, 3.2.1

[40] Ashish Kumar, Toby Buckley, John B Lanier, Qiaozhi Wang, Alicia Kavelaars, and Ilya Kuzovkin. Offworld gym: open-access physical robotics environment for real-world reinforcement learning benchmark and research. *arXiv preprint arXiv:1910.08639*, 2019. 3.2.2

[41] Aviral Kumar, Aurick Zhou, G. Tucker, and S. Levine. Conservative Q-Learning for Offline Reinforcement Learning. *ArXiv*, abs/2006.04779, 2020. 2.1, 2.2

[42] Aviral Kumar, Anika Singh, Stephen Tian, Chelsea Finn, and Sergey Levine. A workflow for offline model-free robotic reinforcement learning. In *CoRL*, 2021.

2.2, 2.7

[43] Aviral Kumar, Joey Hong, Anika Singh, and Sergey Levine. When should we prefer offline reinforcement learning over behavioral cloning? *ArXiv*, abs/2204.05618, 2022. 2.1

[44] Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. When should we prefer offline reinforcement learning over behavioral cloning? *arXiv preprint arXiv:2204.05618*, 2022. 2.2, 2.3, 2.5.2

[45] Vikash Kumar and Emanuel Todorov. Mujoco haptix: A virtual reality system for hand manipulation. In *International Conference on Humanoid Robots*, pages 657–663. IEEE, 2015. 3.3.3

[46] Sascha Lange, Thomas Gabel, and Martin A. Riedmiller. Batch Reinforcement Learning. In *Reinforcement Learning*, volume 12. Springer, 2012. 2.1

[47] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37 (4-5):421–436, 2018. 3.2.3

[48] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020. 2.1

[49] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015. 2.1

[50] Ziyuan Liu, Wei Liu, Yuzhe Qin, Fanbo Xiang, Minghao Gou, Songyan Xin, Maximo A Roa, Berk Calli, Hao Su, Yu Sun, et al. Ocrtoc: A cloud-based competition and benchmark for robotic grasping and manipulation. *IEEE Robotics and Automation Letters*, 7(1):486–493, 2021. 3.2.2

[51] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, pages 879–893. PMLR, 2018. 3.1, 3.2.3

[52] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021. 2.2, 2.3, 2.4, 2.5.2

[53] Elman Mansimov and Kyunghyun Cho. Simple nearest neighbor policy method for continuous control tasks, 2018. URL https://openreview.net/forum?id=ByL48G-AW. 3.5.2

[54] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 2.1

[55] Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020. 2.1, 2.2, 2.4

[56] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022. 2.3, 3.5.1

[57] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, Jan Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018. 2.2, 2.5.2

[58] Jyothish Pari, Nur Muhammad Shafiullah, Sridhar Pandian Arunachalam, and Lerrel Pinto. The surprising effectiveness of representation learning for visual imitation. *arXiv preprint arXiv:2112.01511*, 2021. 3.5.2

[59] Simone Parisi, Aravind Rajeswaran, Senthil Purushwalkam, and Abhinav Kumar Gupta. The unsurprising effectiveness of pre-trained vision models for control. In *ICML*, 2022. 3.5.1

[60] Daniel Pickem, Paul Glotfelter, Li Wang, Mark Mote, Aaron Ames, Eric Feron, and Magnus Egerstedt. The robotarium: A remotely accessible swarm robotics research testbed. In *International Conference on Robotics and Automation*, pages 1699–1706. IEEE, 2017. 3.2.2

[61] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *International Conference on Robotics and Automation*, pages 3406–3413. IEEE, 2016. 3.2.3

[62] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988. 2.1, 2.4

[63] Rafael Figueiredo Prudencio, Marcos R.O.A. Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *ArXiv*, abs/2203.01387, 2022. 2.7

[64] Rongjun Qin, Songyi Gao, Xingyuan Zhang, Zhen Xu, Shengkai Huang, Zewen Li, Weinan Zhang, and Yang Yu. Neorl: A near real-world benchmark for offline reinforcement learning. *ArXiv*, abs/2102.00714, 2021. 2.1

[65] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

3.5.1

[66] Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. In *L4DC*, 2021. 2.2

[67] Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Visual adversarial imitation learning using variational models. In *NeurIPS*, 2021. 2.2

[68] Aravind Rajeswaran, Igor Mordatch, and Vikash Kumar. A Game Theoretic Framework for Model-Based Reinforcement Learning. In *ICML*, 2020. 2.1

[69] Siddharth Reddy, Anca D Dragan, and Sergey Levine. Sqil: Imitation learning via reinforcement learning with sparse rewards. *arXiv preprint arXiv:1905.11108*, 2019. 2.2

[70] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust Region Policy Optimization. *CoRR*, abs/1502.05477, 2015. 2.1

[71] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347, 2017. 2.1

[72] Guna Seetharaman, Arun Lakhotia, and Erik Philip Blasch. Unmanned vehicles come of age: The darpa grand challenge. *Computer*, 39(12):26–29, 2006. 3.1, 3.2.1

[73] Takuma Seno and Michita Imai. d3rlpy: An offline deep reinforcement learning library. *arXiv preprint arXiv:2111.03788*, 2021. 2.4, 3.5.2, A.4

[74] Pratyusha Sharma, Lekha Mohan, Lerrel Pinto, and Abhinav Gupta. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. In *Conference on Robot Learning*, pages 906–915. PMLR, 2018. 3.2.3

[75] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*, 2021. 2.3

[76] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362:1140–1144, 2018. 2.1

[77] Avi Singh, Albert Yu, Jonathan Yang, Jesse Zhang, Aviral Kumar, and Sergey Levine. Cog: Connecting new skills to past experience with offline reinforcement learning. *arXiv preprint arXiv:2010.14500*, 2020. 2.1, 2.2, 2.2

[78] Wen Sun, J Andrew Bagnell, and Byron Boots. Truncated horizon policy search: Combining reinforcement learning & imitation learning. *arXiv preprint arXiv:1805.11240*, 2018. 2.2

[79] Yu Sun, Joe Falco, Máximo A Roa, and Berk Calli. Research challenges and progress in robotic grasping and manipulation competitions. *IEEE Robotics and Automation Letters*, 7(2):874–881, 2021. 3.2.2

[80] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. 3.1

[81] Oriol Vinyals, Igor Babuschkin, Wojciech Marian Czarnecki, Michaël Mathieu, Andrew Joseph Dudzik, Junyoung Chung, Duck Hwan Choi, Richard W. Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, pages 1–5, 2019. 2.1

[82] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018. 3.1

[83] Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. *Advances in Neural Information Processing Systems*, 33:7768–7778, 2020. 2.1, 2.2

[84] Yifan Wu, George Tucker, and Ofir Nachum. Behavior Regularized Offline Reinforcement Learning. *CoRR*, arXiv:1911.11361, 2019. 2.2

[85] Denis Yarats, David Brandfonbrener, Hao Liu, Michael Laskin, P. Abbeel, Alessandro Lazaric, and Lerrel Pinto. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. *ArXiv*, abs/2201.13425, 2022. 2.1, 2.1

[86] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pages 1094–1100. PMLR, 2020. 3.1

[87] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y. Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *ArXiv*, abs/2005.13239, 2020. 2.2

[88] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine,

and Chelsea Finn. Combo: Conservative offline model-based policy optimization. In *NeurIPS*, 2021. 2.2

[89] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5628–5635. IEEE, 2018. 2.2

[90] Tony Z Zhao, Jianlan Luo, Oleg Sushkov, Rugile Pevceviciute, Nicolas Heess, Jon Scholz, Stefan Schaal, and Sergey Levine. Offline meta-reinforcement learning for industrial insertion. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6386–6393. IEEE, 2022. 2.1

[91] Wenxuan Zhou, Sujay Bajracharya, and David Held. Plas: Latent action space for offline reinforcement learning. *arXiv preprint arXiv:2011.07213*, 2020. 2.1

[92] Yuke Zhu, Josiah Wong, Ajay Mandlekar, and Roberto Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020. 3.1

[93] Konrad Zolna, Alexander Novikov, Ksenia Konyushkova, Caglar Gulcehre, Ziyu Wang, Yusuf Aytar, Misha Denil, Nando de Freitas, and Scott Reed. Offline learning from demonstrations and unlabeled experience. *arXiv preprint arXiv:2011.13885*, 2020. 2.2