Deep RL at Scale: Sorting Waste in Office Buildings with a Fleet of Mobile Manipulators

Alexander Herzog*†, Kanishka Rao*‡, Karol Hausman*‡, Yao Lu*‡, Paul Wohlhart*†, Mengyuan Yan†, Jessica Lin†, Montserrat Gonzalez Arenas‡, Ted Xiao‡, Daniel Kappler†, Daniel Ho†, Jarek Rettinghouse† Yevgen Chebotar‡, Kuang-Huei Lee‡, Keerthana Gopalakrishnan‡, Ryan Julian‡, Adrian Li†, Chuyuan Kelly Fu†, Bob Wei†, Sangeetha Ramesh†, Khem Holden‡, Kim Kleiven†, David Rendleman‡, Sean Kirmani†, Jeff Bingham†, Jon Weisz†, Ying Xu†, Wenlong Lu†, Matthew Bennice†, Cody Fong†, David Do†, Jessica Lam†, Noah Brown‡, Mrinal Kalakrishnan†, Julian Ibarz‡, Peter Pastor†, Sergey Levine‡
*Authors with equal contribution †Everyday Robots ‡Robotics at Google

***Abstract*—We describe a system for deep reinforcement learning of robotic manipulation skills applied to a large-scale real-world task: sorting recyclables and trash in office buildings. Real-world deployment of deep RL policies requires not only effective training algorithms, but the ability to bootstrap real-world training and enable broad generalization. To this end, our system combines scalable deep RL from real-world data with bootstrapping from training in simulation, and incorporates auxiliary inputs from existing computer vision systems as a way to boost generalization to novel objects, while retaining the benefits of end-to-end training. We analyze the tradeoffs of different design decisions in our system, and present a large-scale empirical validation that includes training on real-world data gathered over the course of 24 months of experimentation, across a fleet of 23 robots in three office buildings, with a total training set of 9527 hours of robotic experience. Our final validation also consists of 4800 evaluation trials across 240 waste station configurations, in order to evaluate in detail the impact of the design decisions in our system, the scaling effects of including more real-world data, and the performance of the method on novel objects.**

I. Introduction

Real-world robotic manipulation problems require the integration of a range of components, including visual perception, planning, and control. The design and integration of these components, and the abstractions needed to make them work together, often present a challenge for real-world deployment of robotic systems in open-world settings. End-to-end learning offers an appealing alternative: by optimizing the performance of a robotic learning system directly on the final objective of the task, and learning this task directly from visual observations, we can in principle sidestep these challenges and devise robotic manipulation strategies that are fully optimized for the real-world task that they are intended to solve. However, while general-purpose reinforcement learning algorithms can in principle provide such functionality, in practice they suffer from their own set of challenges, including difficulties associated with gathering suitable datasets, generalization, and overall system design. While a broad range of robotic learning methods have been proposed, it remains unclear how to devise an end-to-end learning system that can scale to realistic real-world tasks. In this paper, we approach this system design problem in the context of a complex real-world problem that requires effective and generalizable manipulation strategies: sorting trash and recyclables in office building waste bins. This task naturally provides a broad range of objects for goal-directed manipulation, and serves as a lens to examine how we can design effective and generalizable robotic reinforcement learning pipelines for the real world. The central question in this paper is: how can the various tools in the deep reinforcement learning toolbox be put together to address such a complex and diverse real-world task? Prior works have proposed a variety of tools of this sort, including the use of simulation to overcome the high sample complexity requirements of reinforcement learning from scratch [37, 54], the use of prior data [46, 21, 29], off-policy or offline reinforcement learning strategies [27], large-scale collective learning involving fleets of multiple robots [17, 18], and the use of priors and inductive biases [35, 43, 13]. However, each of these approaches has

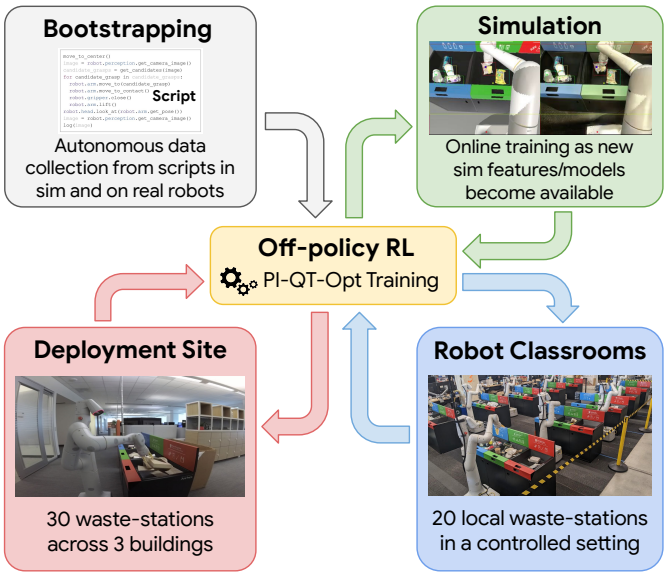

Fig. 1: Overview of our data flywheels that we operated over 24 months: We bootstrap the initial policy from scripts in simulation and on real robots (grey), re-train the policy in simulation as needed (green), deploy the latest policy weekly to a local setup of 20 robots sorting 20 waste stations on random waste-scenes and scenes encountered in the deployment site (blue), and deploy to 23 robots operating in 3 different buildings sorting 30 waste stations (red).

their own strengths and weaknesses. Simulated training is problematic if we expect to encounter a broad range of real-world situations, as unless we can characterize this range of settings in advance, we might fail to generalize. Collective learning from real-world data gathered by multiple deployed robots provides a very powerful method for gathering data that is representative of what the robot actually encounters in the real-world, but training from scratch in the real world presents a major exploration challenge, and relying entirely on robot-collected data can make generalization challenging. Incorporating prior knowledge from other sources, such as computer vision datasets or data from the Internet, can provide a major boost to generalization, but it remains unclear how to most effectively integrate it into an end-to-end trained system. Our final system, which we call RL@Scale or RLS in short, makes use of all three components: real-world training is bootstrapped using simulation, collective learning by a fleet of 20 robots is used to collect representative experience in the real world, and a visual masking approach is further used to incorporate prior knowledge about object classes and appearance from computer vision datasets to additionally boost generalization to new objects. Our extensive empirical investigation shows the importance of each component, and provides a large-scale case study of how end-to-end deep RL can be deployed for a practical and relevant real-world task at scale.

Our evaluation requires the robot to sort trash and recyclables: the robot locates bins with recyclables, approaches them, and sorts their content by type to minimize contamination (i.e., moving all compostables, recyclable plastic, and trash into their respective containers). This task requires generalization to a wide variety of possible objects, as well as the ability to pick up and identify those objects so as to place them in the right container. The objects that the robot might encounter will include difficult-to-simulate properties, such as in the case deformable chip bags, or might be particularly hard to grasp, such as in the case of large boxes. They will also vary over time and across locations. Thus, this task presents a particularly challenging manipulation scenario, but also one that provides a great testbed for scalable robotic learning.

The contribution of our work is a system for real-world end-to-end deep reinforcement learning, RLS, for a complex robotic manipulation task. While our policies are trained end-to-end, they incorporate additional knowledge via object masks trained on computer vision datasets, and are bootstrapped from simulation to overcome the exploration challenge. We show that these decisions lead to excellent performance in the real world. While the individual components of this system have been explored in various prior works, we focus on their integration into a complete robotic learning system at scale: we believe that our method is the first to be applied at such a large scale to a realistic real-world task, and therefore serves as a valuable case study to understand the considerations and tradeoffs in the design of end-to-end robotic learning systems. We present extensive ablations and comparisons to study the individual components of our system, and show that our best-performing design can sort 84.35% of misplaced objects on challenging waste scenarios. We demonstrate dexterous robotic manipulation of waste that emerge from the end-to-end formulation of our policy. When deployed at office buildings, our fleet of mobile manipulators further reduce contamination of waste stations filled by office workers unrelated to the project as part of their day-to-day by up to 53%.

II. Related Work

Advances in reinforcement learning algorithms [1, 55, 2] have enabled machine learning systems that can play games [34, 45], control robots [36, 17, 37, 5, 18], and perform a variety of tasks from chip design to drug discovery [33, 39]. However, real-world deployment of RL systems, particularly in robotics, presents a number of major challenges [8].

First, deep RL algorithms typically require a large number of samples, especially when learning large, image-based policies [34]. Because of this, a common choice is to employ simulation for training [42, 32, 14, 10]. While policies trained in simulation can work well in relatively constrained real-world settings in a laboratory [32, 37, 6], simulated training alone can be insufficient for robotic systems that must generalize to a wide variety of real-world environments, where the range and variability of objects and settings might exceed the variability seen in the simulator. In our proposed robotic RL system, we employ simulated data generation, but combine it with real-world data that is collected autonomously through a variety of policy bootstrapping approaches, and find that this hybrid design allows our system to generalize well in the real world.

Second, RL agents that learn complex tasks from scratch must typically spend considerable time on exploration to discover effective behaviors [3, 38, 49], particularly with simple sparse reward functions that merely indicate success or failure at the task and provide little guidance for how to improve before a successful behavior is discovered. To avoid this challenge, many prior works have explored imitation learning as an alternative to RL [53, 31, 44, 15], or as a way to supplement RL [50, 40, 29] in robotics. In our system, we also sidestep exploration, but we find that we can do this effectively with a combination of relatively simple scripted exploration policies, bootstrapping from simulation, and the use of multi-task training to learn simple tasks as a stepping stone to more complex ones. While prior works have studied each of these methods individually [17, 40, 47], our system combines all of these components into a complete robotic manipulation system that tackles a complex real-world task.

Third, RL-trained policies, like any machine learning model, are vulnerable to distributional shift: when the conditions at deployment-time do not match the conditions seen in training, a learned model will underperform. RL provides an appealing solution to this problem: as the agent experiences new domains, it can simply keep training and continue to adapt [16]. While this observation is not new, we show in our work how a complete RL-based robotic manipulation system

can benefit from this capability and get better as it observes more real-world data.

Our work is related to a number of large-scale learning-enabled robotic manipulation systems that have been proposed in prior work. Many of these systems have been shown to learn behaviors that are physically complex [52, 37] or exhibit good generalization [17, 40, 5, 30, 18, 29, 26], but their evaluations are typically confined to laboratory settings, or else to tasks such as locomotion or navigation that do not require manipulating a wide variety of objects [11, 28, 20]. In contrast, our aim is to develop and evaluate a system that can be deployed at scale on a realistic object manipulation task, with a real-world deployment and a quantitative evaluation that evaluates on scenes replicated from this deployment.

III. Preliminaries

Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, p_0, \gamma)$ define an MDP, where $\mathcal{S}$ and $\mathcal{A}$ are state and action spaces, P is a state-transition probability function, R is a reward function, p_0 is an initial state distribution, γ is a discount factor. We use a sparse reward function that assigns a reward of 1.0 if the task was accomplished successfully at the end of the episode, and 0.0 otherwise. The goal of reinforcement learning is to find a policy $\pi(\mathbf{a}|\mathbf{s})$ that maximizes the expected discounted reward over trajectories induced by the policy, $\mathbb{E}_\pi[R(\tau)]$, where $\mathbf{s}_0 \sim p_0, \mathbf{s}_{t+1} \sim P(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ and $\mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t)$.

To train our end-to-end sorting policy, we use PI-QT-Opt [26], a combination of QT-Opt [17] and a predictive information representation auxiliary [23, 25, 26].

QT-Opt is a value-based RL method that learns a Q-function by minimizing the Bellman error, and obtains the policy by running a stochastic optimization on the learned Q-function using cross-entropy method (CEM) [41], a zeroth-order optimizer that finds $\arg\max_{\mathbf{a}_t} Q_\theta(\mathbf{s}_t, \mathbf{a}_t)$ for the learned Q-function $Q_\theta(\mathbf{s}_t, \mathbf{a}_t)$. Importantly, for CEM to be effective in this setting, the actions generated by the RL agent should be in distribution of the actions sampled by the CEM in the Bellman update equation described below. To learn the Q-function, QT-Opt optimizes the following objective: $\mathcal{L}(\theta) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \sim p(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})} \big[D(Q_\theta(\mathbf{s}_t, \mathbf{a}_t), Q_T(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})) \big]$, where $Q_T(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \max_{\mathbf{a}'} Q_\theta(\mathbf{s}_{t+1}, \mathbf{a}')$ is a target Q-value, γ is a discount factor, and D is a divergence metric. Since the cumulative reward for each episode is always between 0.0 and 1.0, the Q-values can be treated as probabilities (formally, denoting the probability that the robot will succeed at the task), and thus we can use the cross-entropy loss for D. The target value $\max_{\mathbf{a}'} Q_\theta(\mathbf{s}_{t+1}, \mathbf{a}')$ is computed by running the same CEM optimization as we use for the policy action selection. In this work, we use the version of QT-Opt that includes predictive information called PI-QT-Opt [26]. We provide the preliminaries of predictive information in the appendix.

IV. Task and System Overview

Our aim is to develop a deep RL system that can be applied to a real-world waste sorting task, exhibits robust performance and good generalization, and can integrate both autonomously collected real-world experience and bootstrapping from simulation and computer vision systems. In this section, we first describe the waste sorting task we tackle, and then provide an overview of the different components of our system.

A. Waste Sorting Task

We situate our deep RL system in the context of a waste sorting task, which requires separating trash, compostables, and recyclables in waste bins. Since office buildings are created to serve a large number of people, they usually have multiple trash stations distributed throughout different floors, resulting in broad range of environments and potential trash sorting scenarios. Each sorting station contains three bins that correspond to recyclables, compost and landfill. Office workers unrelated to the project use these stations in their regular day-to-day routines. Usually, they make mistakes and misplace their trash into incorrect bins resulting in bin contamination. Our goal is to deploy an autonomous robot to reduce the resulting contamination (i.e., move all the compostables, recyclables, and landfill waste items into their respective bins). We present the visual depiction of the task in Fig. 2. This task presents a number of challenges for a robotic manipulation system: (i) the robots must be able to sort waste at the stations in many different locations; (ii) the manipulation controllers must generalize effectively to previously unseen objects, since people will deposit new and unexpected items into the bins; (iii) many of the objects that are encountered are especially difficult to manipulate, including deformable or complex shaped objects. Examples are shown in Fig. 10.

We assign a reward to an episode when the robot picks up a misplaced object, moves it to the correct bin, and terminates the episode. After termination, we open the robot gripper and observe which bin the object landed in. This reward structure results in an additional partial observability challenge, since the reward depends on where the object was grasped from (i.e., the robot can't simply "cheat" by picking up an already correctly sorted object from its bin and putting it back down), which motivates our use of recurrent architectures with memory as discussed in Sec V.

Iterations of our system, RLS, were deployed in three office buildings with 30 waste stations over the course of two years at various points in its development, resulting in 3803 autonomous station visits by the robots. Although the particular design of the system evolved over this time, this experience combined with more structured data collected in controlled settings, as well as simulated data, were included in the components of our final design, and the final evaluation scenarios used in our experiments were modeled on the challenges observed during deployment. Fig. 6 shows a small subset of scenarios encountered by the robot during deployment.

B. Robotic System Overview

We use a fleet of mobile manipulators with 7-degrees-of-freedom arms and parallel jaw grippers as shown in Fig. 2. For

Fig. 2: The experimental platform. *left*: Our mobile manipulator with a 7 degree-of-freedom (DoF) arm and a parallel jaw gripper. *right*: The sorting task demonstrated by an example: A compostable food container (red box) is misplaced in the landfill tray. Once the robot arrives at its initial state in front of the waste station with the arm above the station, it executes a trained or scripted policy that identifies misplaced objects and moves them to the correct bin. In the case of this example, the robot would receive a reward for moving the food container into the compost tray (green box).

our problem, the state observations $\mathcal{S}$ for the robot corresponds to a $640 \times 512 \times 3$ RGB image observation from the robot's camera as well as a few proprioceptive signals that include the current tool height as well as the target tool pose and gripper aperture that the robot is moving towards. The action space $\mathcal{A}$ controls the whole body and consists of a target tool position and orientation and gripper aperture as well as target pose of the mobile base. Specifically, our action space consists of 10 dimensions: 3D position and orientation of the end-effector, gripper closedness, x and y position and yaw of the base and an added dimension for whether to terminate, each specified in the frame defined by the robot base and as deltas to the robot's current pose. Our model chooses either an arm or a base motion at 1Hz or it chooses to terminate an episode. The commanded arm targets are mapped to straight-line Cartesian trajectories. Base targets are converted to a trajectory of two half-circles connecting the current and target base poses.

V. RL@SCALE

Our method, RLS, incorporates end-to-end deep RL with data from simulation, data from the real world, and computer vision components to address the waste sorting task described in the previous section. While the basic deep RL methodology underlying our approach largely follows prior work [17, 26], our full robotic learning system must address a number of important challenges that arise both with real-world application of deep RL in general, and with the waste sorting task in particular. First, deep RL requires overcoming exploration challenges that might necessitate impractical amounts of data collection when learning from scratch directly in the real world. We therefore employ several bootstrapping strategies to prime the learning process: we employ scripted policies that attain a low but non-zero success rate at object grasping (not necessarily the full sorting task) to collect initial object interaction data, together with simulated data collection. We also employ a multi-task curriculum learning strategy to bootstrap the complex sorting task with simpler tasks, such as grasping indiscriminately or grasping objects of specific types. This allows us to collect data both in simulation and in the real world, which we can incorporate into an offline RL method based on PI-QT-Opt [17, 26]. To address the partial observability of this manipulation task, we use recurrent models with memory (LSTMs) to represent the value function. Finally, since the waste sorting task requires broad generalization to new objects and semantic determination of the type of waste that each object represents (recyclable, compostable, or landfill), we additionally incorporate inputs from a computer vision system based on ShapeMask [22] to additionally boost the generalizability of our method, while retaining the benefits of end-to-end policy learning. We describe the bootstrapping procedure, data collection process, RL approach, and the vision system integrating in the subsequent sections.

A. Bootstrapping

In order to bootstrap our real-world RLS policy, we use three different and complementary sources of data: scripted policies, simulation, and multi-task RL objectives. We start this process by employing scripted policies in simulation and in the real world. The data collected from these is used to train RetinaGAN [10], which transforms simulated images to look more realistic. Equipped with this sim-to-real transfer tool we deploy policies learned in simulation in the real-world and run them side by side with scripted real-world policies. Since both of these policies encounter only occasional full-sort successes, we further employ multi-objective RL to enable faster bootstrapping. Below we describe these different components in more detail.

Scripted policy. We start by implementing a simple scripted indiscriminate grasping policy (i.e., a policy that lifts any object) in simulation and in the real world. In each episode, the scripted policy would randomly detect an object using an off-the-shelf object detector (or in simulation ground-truth), plan the grasp pose, and generate the trajectories to reach the object. Even though the initial success rate of this policy is around 20%, it is enough to bootstrap simulation and start collecting data on the real robots. We start collecting data in a laboratory environment that we refer as *robot classrooms*, where we deploy the scripted policy on 20 robots that are located in front of sorting stations, where they continuously perform the task of indiscriminate grasping.

Sim-to-real. Our simulation policy is initially bootstrapped using the same scripted policy as deployed in the real world. Once it gathers enough successes, we start reinforcement learning training in simulation. The simulation policy is trained with PI-QT-Opt [26]. Once a simulation policy is trained, to enable sim-to-real transfer, we transform simulated images to look more realistic using RetinaGAN [10], which is trained on real and simulation camera images. Even though this initial sim-to-real policy has a relatively low performance, it is good enough to simultaneously start deploying it in the *robot classrooms* to provide an additional source of real-world data besides the scripted policy.

Multi-task RL. Once we have good performance of indiscriminate grasping in both simulation and the classrooms,

we start to encounter very occasional successes of the more complex tasks such as displacing an object (moving it from on tray to another) or even sorting (moving it to the correct target bin). Since the reward signals for the sorting task are still too rare, we introduce the final bootstrapping step that allows us to generate a large number of successful sorting examples. We take advantage of multi-task reinforcement learning where we introduce de-facto a curriculum of various task difficulties, the most difficult of which is the task of sorting waste. In particular, we train a sorting policy using a multi-task training strategy described in MT-Opt [18]. We therefore devise a total of 15 tasks, including "indiscriminate grasping", "grasp recyclable", "grasp compostable", "grasp landfill", "indiscriminate from recycling", "indiscriminate from compost", "indiscriminate from landfill", "grasp misplaced recyclable", "grasp misplaced compostable", "grasp misplaced landfill", "displace object", "sort recyclable", "sort compostable", "sort landfill" and finally "sort". Since the easier tasks experience more successes initially, they learn faster and lead to more successes of the more difficult tasks, which in turn bootstrap even more difficult tasks and so on. Once the multi-task policy has converged, we switch to only collecting data for the sort task and only training on the sort task for further improvements of the policy.

B. Real-World Challenges and Data Collection

Next, we go into detail on various design choices that were made to address the real-world challenges of collecting a large-scale dataset with autonomous robots sorting waste.

Scripted exploration. As described in the previous section, we start our data collection by running a scripted grasping policy that succeeds some of the time using off-the-shelf object detectors and grasp planning techniques. The goal of this policy is to generate enough data that, together with simulated data, will bootstrap the real-world RL policy. To accomplish this, the scripted policy needs to fulfill two requirements. First, it needs to encounter occasional successes that can be used to bootstrap the more powerful RL policy, which is described in Sec. V-A. The second, more subtle requirement is that the distribution of actions for the scripted policy should be similar to that of the future RL policy, to provide sufficient coverage to support learning a more optimal policy with RL. To address this, we propose an action conversion mechanism and, together with the underlying script, we refer to this component as *scripted exploration*. To this end, the scripted policy we use for exploration first generates waypoints to reach for randomly selected objects proposed by an off-the-shelf object detector. It then uses these waypoints as "attractors" for a pseudo-value-function that is then fed to the same CEM process that we use to select actions with RL. That is, instead of optimizing the action via CEM to maximize its Q-value, we optimize it with CEM to minimize the distance to the waypoint, using the same proposal distribution that is used for running the RL policy. Since the stochastic CEM optimizer is imperfect, this leads to the same noise profile in both the exploration policy and the final RL policy. We present the details of the scripted exploration algorithm in the appendix.

Autonomous data collection. Deploying robots for extended periods of autonomous operation and data collection, both in *robot classrooms* and in the real world, is central to our data collection strategy. To prevent damage to the robot and its environment, while still allowing the robot to make contact with the world and manipulate it, we employ the following three strategies: (1) When sampling potential actions for the arm and the base during the CEM phase of the algorithm, we restrict the samples for arm poses to a box spanning the waste trays and the area above and the orientation of the gripper to not be pointing upwards. CEM samples for movements of the base are constrained to a rectangular area in front of the waste station.[1] (2) We employ a controller that executes commands only in a *best-effort* manner, meaning that trajectories are only executed to the point where any part of the robot (except the fingers) would collide with a voxel map representation of robot's direct environment, reconstructed from the robots depth sensors. (3) All motion is interrupted when a force threshold of 7N is exceeded at the wrist.

C. Reinforcement Learning Flywheel

Equipped with real and simulated data, we use deep RL to train an end-to-end policy that is directly optimized for reducing the contamination of the bins. Similarly to how we train our simulation policy, we leverage PI-QT-Opt [26] to train the final policy on the complete dataset assembled from simulation and real world collection. Deep RL allows us to not only distill the best possible policy out of the bootstrapping data, but also to enable the robot to improve continuously as it interacts with waste stations more and more. We refer to this iterative improvement as a *data flywheel*: a continual process where the robot performs the waste sorting task, gathers more experience, and incorporates this experience back into the RL process to further improve its policy.

Since training and updating the policy after each sample across a large fleet of robots is impractical from a systems engineering perspective, we update the policy and deploy it to the fleet iteratively in batches. For each iteration of the flywheel, we deploy an updated model to the robot fleet and collect a batch of data over a week of operation time. We train our model with PI-QT-Opt using all the data collected so far, including the newly added batch, and deploy the converged model back to the robot fleet and proceed with the next flywheel iteration. Since we train our policy on real robots and we aim to deploy the best possible version of the policy at all times, we use an offline policy evaluation metric called OPC [12] to find the most performant checkpoint of the current flywheel iteration.

D. Neural Network Architecture

In Fig. 3 we present the diagram of the neural network architecture of the Q-function that is learned with PI-QT-Opt. We feed two RGB images to two separate convolutional towers

[1] A visualization of the arm and base workspace constraints is shown in the supplemental material.

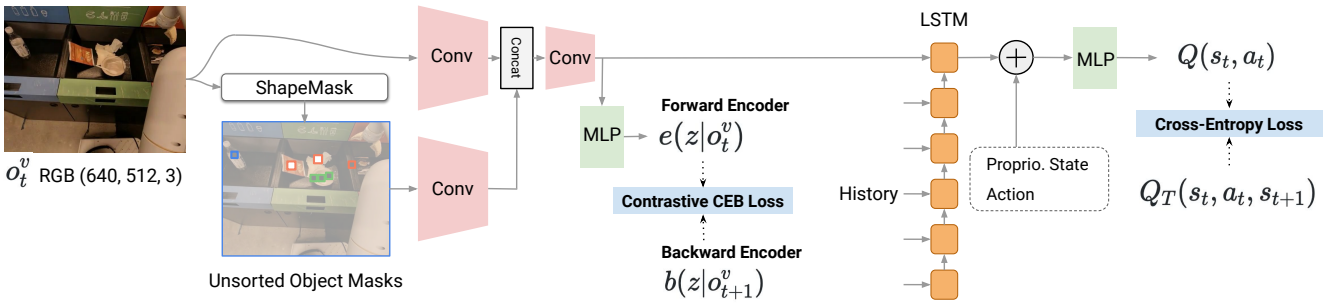

Fig. 3: An overview of the network architecture. We encode RGB camera images and unsorted object masks convolutional layers. Our Q-function considers visual observations in the most recent 6 time steps, but the predictive information (CEB) auxiliary only considers the current image o_t^v for the past X and the next image o_{t+1}^v for the future Y, in order to avoid information overlap between X and Y.

which are later concatenated and processed by another set of convolutional layers. The two images correspond to the current camera image as well as the object mask image that we describe in the next section. After the last convolutional layers, we add a small multi-layer perceptron (MLP) to parameterize the forward encoder for predictive information learning. We use the same convolutional architecture with moving average parameters of that for the current time step to encode the two RGB images of the next time step. With the forward and backward encoders, we enforce the contrastive CEB loss described in Sec. III.

To further aid with the long-horizon task of sorting waste, we enhance our policy by adding a set of LSTM layers to cope with the partial observability of the task. In particular, we employ a PI-QT-Opt-adjusted R2D2 approach [19] to pass the recurrent policy state from data collection to the replay buffer and further to the trainer. Towards the end of the neural network processing, we add proprioceptive information as well as the action that is being scored by the Q-function. These, together with the previous signals are processed by an extra set of MLP layers to output the final Q-value.

E. Boosting Generalization with Pretrained Object Masks

While the simulated and real-world datasets can provide our waste sorting policy a sufficient level of performance to continually collect useful data, diverse real-world sorting scenarios necessitate a very high degree of generalization due to the variety of objects that the robot is likely to encounter. Furthermore, the robot must be able to correctly determine the type of each object (compostable, recyclable, or landfill). Therefore, to further boost generalization, we integrate a pre-trained computer vision model that is further fine-tuned on labeled data for the sorting task.

We use a pre-trained one-stage segmentation model to predict panoptic segmentation masks (a combination of instance and semantic masks). The model consists of an EfficientNet-B1 [48] backbone and separate convolutional heads for object-wise classification, box regression, and mask prediction. We also leverage mask priors to help generalize to different object shapes as in ShapeMask [22]. The backbone is first pretrained on ImageNet [7] through the noisy-student setup [51], and then the entire model, including the mask priors, is finetuned on a separate perception dataset consisting of 30k examples labeled with panoptic masks, all collected from the robot during operations in the same waste sorting task setup. The list of object classes used for semantic segmentation is a superset of the waste objects seen during sorting.

Once the object mask predictor is trained, we incorporate the information it provides into the policy network. To achieve this, we create an extra image with a dot at the center of every object that is currently misplaced. The color of the dot indicates which bin the object should be sorted into. This image is fed to the network as an extra input channel concatenated to the current RGB image as shown in Fig. 3.

This design presents a number of advantages. We incorporate knowledge from pretraining on computer vision datasets, enabling the robots in principle to classify and sort objects that they have not interacted with. Further, it allows the RL policy to focus on the manipulation aspects of the sorting task (e.g., how to singulate and grasp an object) rather than the semantic classification, which can be achieved more efficiently using supervised learning. At the same time, in contrast to more standard approaches that decompose localization, planning and control into separate modules, our deep RL approach is still end-to-end, in the sense that the Q-function still operates on raw images and actions and can use them in whatever way it needs to maximize task performance, while additionally receiving the object mask as a supplementary input.

VI. Real-World Experiments

In our evaluation, we seek to answer the following questions: (1) How does the performance of RLS change with various amounts of data in terms of in-distribution performance as well as generalization to new sorting scenarios? (2) What is the impact of various design decision on the final performance of RLS? (3) What are the overall trends and performance metrics from deploying RLS in real office buildings?

A. Experimental Setup

For data collection in the *robot classrooms*, we use a fleet of 20 robots that continuously sort waste from 20 waste stations (see Fig. 5). We randomly place objects into the waste stations and shuffle them several times throughout the day to increase contamination. After an initial bootstrapping

Fig. 4: Waste scenarios used for evaluations. Top 3 rows show the 9 in-distribution scenarios. The bottom row shows the held-out-scenes, containing objects previously seen neither in the real world nor in simulation, such as the keyboard, banana and face-mask.

Fig. 5: The robot classroom, a controlled setting for repeatable evaluations. 20 robots continuously collect data at 20 waste stations.

phase (see Sec. V-A) we deployed our flywheel and collected 540k episodes in the *robot classrooms* over the course of 4 months. Over the course of the project, we also deployed various iterations of our method to 23 robots servicing 30 waste stations across three operational office buildings, which we refer to as the *deployment site*, which we describe in detail in the appendix. Waste stations at deployment sites are filled with waste entirely by people that ordinarily work in these buildings. We gathered 32.5k episodes from our deployment and added them to the overall training set. Note that data throughput was significantly higher in the *robot classrooms*, where robots could sort waste continually, whereas the available unsorted waste at the *deployment sites* varied drastically over time (particularly during stages of the project that overlapped with the COVID epidemic). For repeatable experimentation, we identified 9 challenging waste scenarios from the deployment site and additional 3 held-out scenes which contain objects that the robot has not interacted with in the training data (see Fig. 4). A waste scenario prescribes how many instances of each object class are placed in which of the three bins before the robot starts sorting. The placement of objects within each bin are randomized. We perform extensive evaluations on the two sets of waste scenarios. Each robot has a maximum of 20 attempts to sort all the objects in each scenario, and the scenarios have between 2 and 9 initially misplaced objects. We report the sorting success rate over 2 rounds for a total of 360 attempts (20 attempts $\times$ 9 scenarios $\times$ 2 rounds).

B. Quantitative Evaluation of RLS

Since our system RLS is deployed at scale over the course of multiple months, including deployment in actual office buildings with constantly changing trash scenarios, it is challenging to perform a clean set of experiments delineating the influence of different components of the system. We therefore use the *robot classrooms* setup for ablations and experiments, using the scenarios described in Sec. VI-A and shown in Fig. 4.

To answer our first question: *How does the performance of RLS change with various amounts of real world data in terms of in-distribution performance as well as generalization to new sorting scenarios?* we perform a set of evaluations where we vary the amount of real-world data included in training the model. In particular, we perform training of RLS in iterations in the *robot classrooms*, where each iteration involves using the policy from the previous iteration to collect data for several days. A new policy is then trained on all the data, including the latest iteration, and if performance improves over the previous policy then it is used as the collection policy for the next iteration. For data collection, we set up scenarios with the same objects as the in-distribution evaluation scenario, but with object numbers and configurations randomized, and we additionally report performance on unseen scenarios. Robots are allowed to autonomously operate with occasional reshuffling of the objects by human operators. All episodes are labeled by humans (via an image labeling interface similar to ones used for other image labeling work) to mark if an object is sorted or not, and these binary labels are directly used as the RL reward.

We present the results in Fig. 7. In-distribution performance is averaged over nine evaluation scenarios, and generalization performance is averaged over three held-out scenes, as described in Sec. VI-A. The policy steadily improves with more real-world data, with 84% of the objects correctly sorted for the final policy. We observe the biggest improvement in the

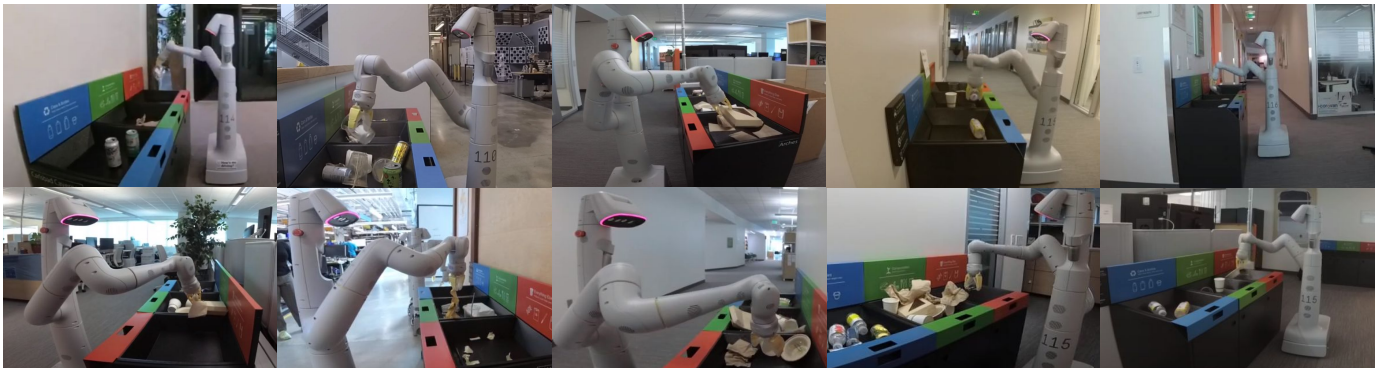

Fig. 6: Example scenes from the deployment sites in 3 different buildings, showing the robot interacting with waste stations used by people over the course of their standard daily routines. Note the high variability in quantity and contamination levels.

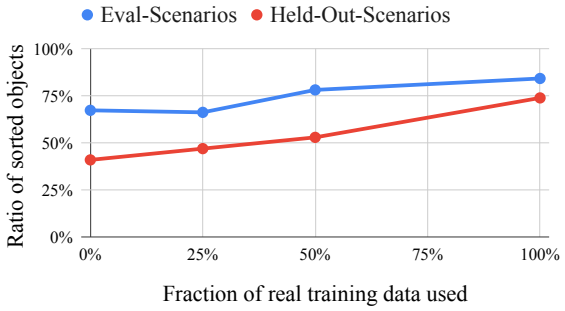

Fig. 7: Waste sorting performance when using 0%, 25%, 50%, and 100% real world training data on the evaluation scenarios (blue) and the held-out scenarios (red). 0% represents training RLS only in simulation. As expected, the performance scales with the amount of training data used, is slightly better on the evaluation scenarios, but also generalizes and improves to excellent performance on the held out scenes.

held-out scenarios when switching from 50% of the data to 100%, indicating that more data has a substantial impact on the generalization of RLS.

C. Qualitative Evaluation of RLS

In order to illustrate the difficulty of some of the encountered sorting scenarios, we present a few examples in Fig. 8. These include a number of contact-rich, difficult-to-accomplish manipulation behaviors. These are difficult not only in terms of manipulation, but also the semantics of the task (e.g., Fig. 8a), where the policy needs to distinguish that the object inside a container is correctly sorted already, while the container itself needs to be moved to a different bin. In Fig. 8b, we show an example with poor lighting, which do not prevent our policy from correctly sorting the object. We also demonstrate an emerging re-grasping behavior of our policy (e.g., Fig. 8c), as well as the ability to cope with very large (Fig. 8d) and very small objects (Fig. 8e).

D. Ablations of RLS and Design Decisions

Next, we study the second question: *(2) What is the impact of various design decision on the final performance of RLS?*. We evaluate the performance of variations of our method on the 9 waste scenarios and report ratios of objects sorted in Fig. 9. We observe the biggest negative impact, 45.66% objects sorted, when we train our policy purely on simulated episodes without including any real data, and when in addition we choose a poor initial policy to bootstrap the training process. If we use the original bootstrap model and no real data, we see a performance of 67.39%. We observe a performance of 58.7% when we remove the object mask, forcing the policy to rely on only the RGB input, indicating the value of introducing auxiliary semantic knowledge from the masks. If instead we remove the LSTM architecture and fall-back to a memory-less architecture we see a performance of 69.57%. Note that for this baseline, we filter out episodes where the robot incorrectly grasped an object that was not misplaced, so the drop in performance is not due only to the inability to determine which objects were already sorted. Finally, our method, RLS, with all design decisions included, sorts 84.35% of the objects. Overall, we see that the design choices made for RLS are important to achieve good performance on a challenging real world task as trash sorting.

E. Evaluating the Real-World Deployment of RLS

We deployed our system, RLS, in three office buildings using a fleet of 23 robots servicing 30 waste stations over the course of 14 months. Based on this deployment, we gathered 32.5k waste sorting episodes and visited the sorting stations a total of 3803 times. The unstructured nature of this task makes it difficult to perform rigorous head-to-head comparisons, but we report some statistics of this deployment below.

First, to provide an intermediate evaluation between the more structured *robot classrooms* and the fully unstructured deployment, we picked one waste station in each building and set up a scene analogous to our *robot classrooms* evaluation scenario in each one. In this case, the results show a negligible difference, with a 92.7% success rate in *robot classrooms* and 93% in the office buildings, confirming that our policy generalizes to different locations.

Next, we fully deploy RLS "in the wild," where our robot fleet needs to sort previously unseen waste in novel waste sorting scenarios that are created by the occupants of these buildings. We deploy our system over three time periods in

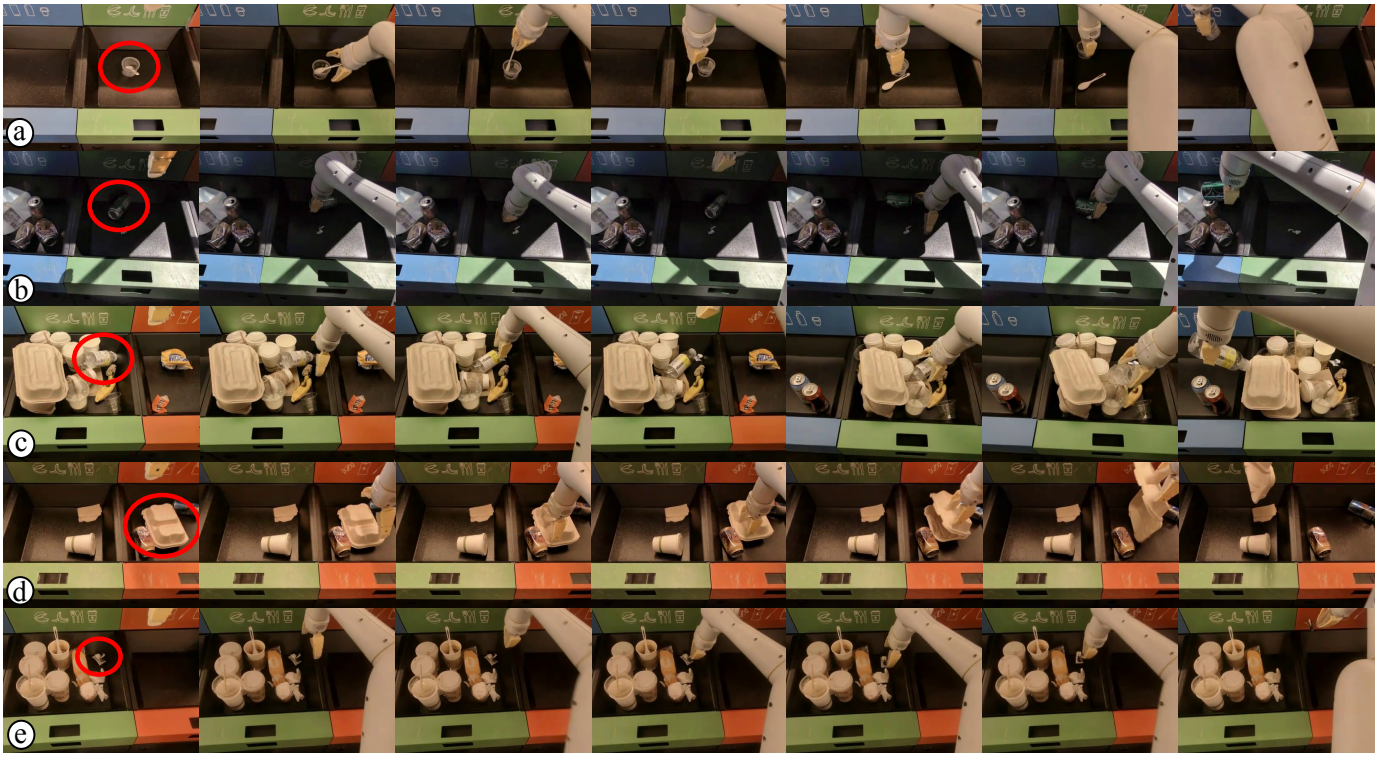

Fig. 8: Five successful sorts illustrating the dexterity of the learned policy: (a) Removing the compostable spoon (red circle) and correctly sorting the yogurt cup. (b) Sorting a soda can (red circle) under poor lighting. Note how at first the can is occluded and the robot lifts its gripper to make it visible again. (c) Sorting a plastic bottle (red circle) after a few re-grasps. (d) Sorting a large lunch box. (e) Pinching and sorting a small candy wrapper (red circle).

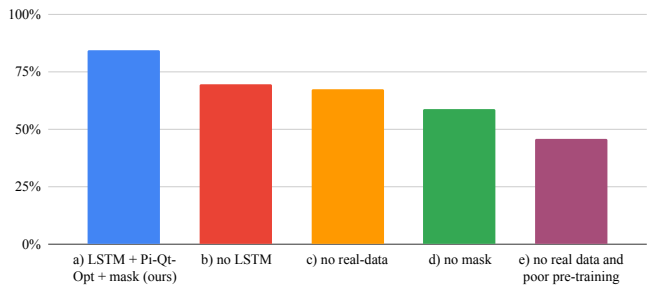

Fig. 9: Ratio of sorted objects of ablations of our approach. a) our method, RLS b) our method without LSTM c) trained with sim data only c) without mask-conditioning d) trained with sim data only and a poorly performing bootstrap model.

Deployment days (dates)	Waste station visits	Contamination reduction
165 (11/2021-4/2022)	15	53%
160 (8/2021-2/2022)	51	52%
104 (5/2022-8/2022)	277	39%

TABLE I: Statistics from the three deployments in three office buildings that span $\sim$ 2 years. The first two rows showcase fewer waste station visits because of the global pandemic, which resulted in lower amounts of trash in the waste sorting stations in the office buildings.

2021 and 2022, each one of them spanning between 100 and 170 days. Because of the global pandemic, the first two deployments experience much lower amounts of waste in the sorting stations, and therefore fewer waste station visits. We summarize our results in Table I.

The contamination reduction in Table I is calculated as a ratio between the *weight* of all the initially misplaced objects in the bins and the weight of the misplaced objects at the end of RLS runs. We measure the contamination reduction to initially be 53% during the first deployments, which experienced lower traffic because of the pandemic. The contamination reduction drops to 39% once there are more people in the office buildings, when our system experiences much more diverse sorting scenarios. To visualize the diversity of the encountered scenarios, we show some examples in Fig. 10. These include overflowing bins (last row, left column), objects that are too large for our robot to possibly grasp, such as keyboard packaging (first row, middle column), and tightly packed objects that are difficult to manipulate (last row, last column). These examples illustrate the gap between the real-world "in the wild" distribution of sorting scenarios and the distribution experienced in the *robot classrooms*, where RLS achieves a high sorting performance. However, in all cases, the RLS policy led to significant reductions in contamination, resulting in a considerable reduction in the amount of waste that would otherwise have been improperly sent to a landfill or contaminated a recycling batch.

Fig. 10: Example waste scenarios observed during site deployment. We observe a significantly wider range of objects such as various packaging materials, large plastic wrappers, very small torn up papers/candy wrappers, glass bottles, spilled liquids, and perished food. Also, we observe much more diverse object configurations, trays are densely packed and objects are carefully piled and possibly overhanging.

VII. Limitations and Conclusions

In this paper, we presented a robotic manipulation system based on reinforcement learning that sorts waste at waste stations in office buildings, described a large-scale deployment of this system, and presented a quantitative evaluation based on the experience from this deployment. Our approach aims to illustrate how deep RL methods can be integrated into a robotic manipulation system that is feasible to deploy in the real world and able to benefit from real-world data, simulation, and improved generalization from computer vision components. The policy is bootstrapped from a combination of simulated data and experience collected with a scripted policy, from where it is further improved through additional data collection in *robot classrooms* and deployment. The training process employs a curriculum that integrates multi-task learning with simpler tasks to bootstrap the complete sorting task, uses a memory-based LSTM architecture, and integrates additional inputs from an object mask trained on labeled data to boost generalization. We ablate the individual components of our system and evaluate how its performance improves with more data, including on held-out test scenarios.

One limitation of our system is that it still relies heavily on experience gathered in the controlled *robot classroom* settings, rather than entirely using data from real-world deployments. We found the deployments to be highly variable, and also harder for obtaining a steady volume of useful experience, as real-world waste was deposited into the waste stations at different rates during the day and week, resulting in lower data collection throughput than we could obtain in the *classrooms*. We hope that the demonstration of large-scale real-world robotic RL presented in this paper can serve as a prototype for future works that will address this and other future challenges and deploy complete robotic manipulation systems that incorporate RL to enable real-world improvement and generalization.

Acknowledgements

The authors would like to thank Mohi Khansari, Cameron Tuckerman, Stanley Soo, Benjie Holson, Justin Vincent, Mario Prats, Thomas Buschmann, Michael Quinlan, Yunfei Bai, Josphine Simon, Jarrett Lee, Kalpesh Kuber, Meghha Dhoke, Christian Bodner and Russell Wong for their help and support in various aspects of the project.

References

[1] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. Deep Reinforcement Learning: A Brief Survey. *IEEE Signal Processing Magazine*, 2017.

[2] A. T. Azar, A. Koubaa, N. Ali Mohamed, H. A. Ibrahim, Z. F. Ibrahim, M. Kazim, A. Ammar, B. Benjdira, A. M. Khamis, I. A. Hameed, and G. Casalino. Drone Deep Reinforcement Learning: A Review. *Electronics*, 2021.

[3] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. *Advances in Neural Information Processing Systems*, 2016.

[4] W. Bialek and N. Tishby. Predictive information. *arXiv preprint cond-mat/9902341*, 1999.

[5] S. Cabi, S. Gmez, A. Novikov, K. Konyushova, S. Reed, R. Jeong, K. Zolna, Y. Aytar, D. Budden, M. Vecerik, O. Sushkov, D. Barker, J. Scholz, M. Denil, N. Freitas, and Z. Wang. Scaling data-driven robotics with reward sketching and batch reinforcement learning. In *Robotics: Science and Systems*, 2020.

[6] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox. Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience. In *International Conference on Robotics and Automation*, 2019.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition*, 2009.

[8] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester. Challenges of Real-World Reinforcement Learning: Definitions, Benchmarks and Analysis. In *International Conference on Machine Learning*, 2019.

[9] I. Fischer. The conditional entropy bottleneck. *Entropy*, 22(9):999, 2020.

[10] D. Ho, K. Rao, Z. Xu, E. Jang, M. Khansari, and Y. Bai. RetinaGAN: An Object-aware Approach to Sim-to-Real Transfer. In *International Conference on Robotics and Automation*, 2021.

[11] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning Agile and Dynamic Motor Skills for Legged Robots. *Science Robotics*, 2019.

[12] A. Irpan, K. Rao, K. Bousmalis, C. Harris, J. Ibarz, and S. Levine. Off-Policy Evaluation via Off-Policy Classification. *Advances in Neural Information Processing Systems*, 2019.

[13] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement Learning with Unsupervised Auxiliary Tasks. In *International Conference on Learning Representations*, 2017.

[14] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis. Sim-To-Real via Sim-To-Sim: Data-Efficient Robotic Grasping via Randomized-To-Canonical Adaptation Networks. In *Conference on Computer Vision and Pattern Recognition*, 2019.

[15] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, 2022.

[16] R. Julian, B. Swanson, G. Sukhatme, S. Levine, C. Finn, and K. Hausman. Never Stop Learning: The Effectiveness of Fine-Tuning in Robotic Reinforcement Learning. In *Conference on Robot Learning*, 2021.

[17] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine. Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation. In *Conference on Robot Learning*, 2018.

[18] D. Kalashnkov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman. MT-OPT: Continuous Multi-Task Robotic Reinforcement Learning at Scale. In *Conference on Robot Learning*, 2021.

[19] S. Kapturowski, G. Ostrovski, J. Quan, R. Munos, and W. Dabney. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*, 2019.

[20] A. Kumar, Z. Fu, D. Pathak, and J. Malik. RMA: Rapid Motor Adaptation for Legged Robots. In *Robotics: Science and Systems*, 2021.

[21] A. Kumar, A. Singh, F. Ebert, Y. Yang, C. Finn, and S. Levine. Pre-Training for Robots: Offline RL Enables Learning New Tasks from a Handful of Trials. *arXiv preprint arXiv:2210.05178*, 2022.

[22] W. Kuo, A. Angelova, J. Malik, and T.-Y. Lin. ShapeMask: Learning to Segment Novel Objects by Refining Shape Priors. In *IEEE/CVF International Conference on Computer Vision*, 2019.

[23] K.-H. Lee, I. Fischer, A. Liu, Y. Guo, H. Lee, J. Canny, and S. Guadarrama. Predictive Information Accelerates Learning in RL. *Advances in Neural Information Processing Systems*, 2020.

[24] K.-H. Lee, A. Arnab, S. Guadarrama, J. Canny, and I. Fischer. Compressive Visual Representations. *Advances in Neural Information Processing Systems*, 2021.

[25] K.-H. Lee, O. Nachum, T. Zhang, S. Guadarrama, J. Tan, and W. Yu. PI-ARS: Accelerating Evolution-Learned Visual-Locomotion with Predictive Information Representations. In *International Conference on Intelligent Robots and Systems*, 2022.

[26] K.-H. Lee, T. Xiao, A. Li, P. Wohlhart, I. Fischer, and Y. Lu. PI-QT-Opt: Predictive Information Improves Multi-Task Robotic Reinforcement Learning at Scale. In *Conference on Robot Learning*, 2022.

[27] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, 2020.

[28] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza. Deep drone racing: From simulation to reality with domain randomization. *IEEE Transactions on Robotics*, 2019.

[29] Y. Lu, K. Hausman, Y. Chebotar, M. Yan, E. Jang, A. Herzog, T. Xiao, A. Irpan, M. Khansari, D. Kalashnikov, and S. Levine. AW-Opt: Learning Robotic Skills with Imitation and Reinforcement at Scale. In *Conference on Robot Learning*, 2022.

[30] A. Mandlekar, F. Ramos, B. Boots, S. Savarese, L. Fei-Fei, A. Garg, and D. Fox. Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. In *International Conference on Robotics and Automation*, 2020.

[31] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Mart'in-Mart'in. What Matters in Learning from Offline Human Demonstrations for Robot Manipulation. In *Conference on Robot Learning*, 2021.

[32] J. Matas, S. James, and A. J. Davison. Sim-to-Real Reinforcement Learning for Deformable Object Manipulation. In *Conference on Robot Learning*, 2018.

[33] A. Mirhoseini, A. Goldie, M. Yazgan, J. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, S. Bae, et al. Chip placement with deep reinforcement learning. *arXiv preprint arXiv:2004.10746*, 2020.

[34] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015.

[35] M. Mueller, A. Dosovitskiy, B. Ghanem, and V. Koltun. Driving Policy Transfer via Modularity and Abstraction. In *Conference on Robot Learning*, 2018.

[36] H. Nguyen and H. La. Review of Deep Reinforcement Learning for Robot Manipulation. In *IEEE International Conference on Robotic Computing*, 2019.

[37] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. A. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. M. Zhang. Solving Rubik's Cube with a Robot Hand. *ArXiv*, 2019.

[38] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped dqn. *Advances in Neural Information Processing Systems*, 29, 2016.

[39] M. Popova, O. Isayev, and A. Tropsha. Deep reinforcement learning for de novo drug design. *Science Advances*, 2018.

[40] M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. van de Wiele, V. Mnih, N. Heess, and J. T. Springenberg. Learning by Playing Solving Sparse Reward Tasks from Scratch. In *International Conference on Machine Learning*, 2018.

[41] R. Y. Rubinstein and D. P. Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*, volume 133. Springer, 2004.

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

[43] A. Sax, B. Emi, A. R. Zamir, L. Guibas, S. Savarese, and J. Malik. Mid-Level Visual Representations Improve Generalization and Sample Efficiency for Learning Visuomotor Policies. *arXiv*, 2018.

[44] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-Actor: A Multi-Task Transformer for Robotic Manipulation. In *Conference on Robot Learning*, 2022.

[45] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 2018.

[46] A. Singh, A. Yu, J. Yang, J. Zhang, A. Kumar, and S. Levine. COG: Connecting New Skills to Past Experience with Offline Reinforcement Learning. *arXiv preprint arXiv:2010.14500*, 2020.

[47] L. Smith, J. C. Kew, X. B. Peng, S. Ha, J. Tan, and S. Levine. Legged Robots that Keep on Learning: Fine-Tuning Locomotion Policies in the Real World. In *International Conference on Robotics and Automation*, 2022.

[48] M. Tan and Q. V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *International Conference on Machine Learning*, 2019.

[49] H. Tang, R. Houthooft, D. Foote, A. Stooke, O. Xi Chen, Y. Duan, J. Schulman, F. DeTurck, and P. Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 2017.

[50] M. Vecerík, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. A. Riedmiller. Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards. *CoRR*, 2017.

[51] Q. Xie, E. H. Hovy, M.-T. Luong, and Q. V. Le. Self-training with noisy student improves imagenet classification. *Conference on Computer Vision and Pattern Recognition*, 2019.

[52] A. Yahya, A. Li, M. Kalakrishnan, Y. Chebotar, and S. Levine. Collective Robot Reinforcement Learning with Distributed Asynchronous Guided Policy Search. In *International Conference on Intelligent Robots and Systems*, 2017.

[53] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel. Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation. In *International Conference on Robotics and Automation*, 2018.

[54] W. Zhao, J. P. Queralta, and T. Westerlund. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey. In *Symposium Series on Computational Intelligence*, 2020.

[55] K. Zhu and T. Zhang. Deep Reinforcement Learning Based Mobile Robot Navigation: A Review. *Tsinghua Science and Technology*, 2021.

APPENDIX

A. Scripted exploration

We present the scripted exploration algorithm in Alg. 1. Note that the action length and noise of s_min depend on the initial parameters of CrossEntropyMethod (line 10). In particular, we run only 2 iterations of the method for faster computation. We use the same CrossEntropyMethod parameters with our learned critic (Fig. 3) for inference and training. This ensures that generated action samples from the script (line 1) and our learned critic resemble the same action distribution.

Algorithm 1: Scripted Exploration Policy

1 **Def** `GenerateTargetWaypoints()`**:**
2 **if** *IsSimulation* **then**
3 object_poses $\leftarrow$ simulation ground truth
4 **else**
5 object_poses $\leftarrow$ object detector
6 $w_{approach} \leftarrow$ `Random` (object_poses)
7 $w_{grasp} \leftarrow w_{approach}$ with gripper closed
8 $w_{lift} \leftarrow w_{grasp}$ with lift $20cm$ along z-axis
9 **return** $w_{approach}, w_{grasp}, w_{lift}$
10 **Def** `CrossEntropyMethod(`*critic*`)`**:**
11 Initialize Gaussian, μ, σ
12 Initialize number of iterations, $I = 2$
13 Initialize batch size, $N = 128$
14 Initialize number of elites, $M = 13$
15 **for** _ *in I* **do**
16 $\hat{s}_{1:N} \sim \mathcal{N}(\mu, \sigma^2)$
17 $\hat{d}_{1:N} \leftarrow critic(s_1), \ldots, critic(s_N)$
18 $d_1, s_1, \ldots, d_N, s_N \leftarrow rank(\hat{d}_1, \hat{s}_1, \ldots, \hat{d}_N, \hat{s}_N)$
19 $\mu = \frac{1}{M}\sum_{s_{1:M}} s_i;\ \sigma^2 = \frac{1}{M}\sum_{s_{1:M}} (s_i - \mu)^2$
20 **return** s_1, d_1
21 Initialize waypoint threshold, $d_{threshold}$
22 **for** *w in* `GenerateTargetWaypoints` () **do**
23 $d_{min} \leftarrow$ inf
24 **while** $d_{min} \geq d_{threshold}$ **do**
25 $s, d \leftarrow CEM(\texttt{EuclideanDistance}(w, \cdot))$
26 **if** $d \leq d_{min}$ **then**
27 $d_{min} \leftarrow d$
28 $s_{min} \leftarrow s$
29 `Robot.Move` (s_{min})

B. Predictive information preliminaries

Predictive Information [4], the mutual information between the past and the future, has been an effective auxiliary for large-scale real-world RL [26]. From here on, we will denote the past by X and the future by Y. In an MDP, the past X refers to what has been observed by the agent; the future Y refers what will happen in the future process. Lee et al. [23] argues that a learned representation Z of the predictive information should be compressed with respect to X, based on the observation of Bialek and Tishby [4] that $H(X)$ grows faster than $I(X;Y)$. We follow Lee et al. [23] to learn the representation Z with the Conditional Entropy Bottleneck (CEB) [9], using the same contrastive variational bound:

$$\begin{aligned} CEB &\equiv \min_Z \beta I(X;Z|Y) - I(Y;Z) \\ &\leq \min_Z \mathrm{E}_{x,y,z\sim p(x,y)e(z|x)} \beta \log \frac{e(z|x)}{b(z|y)} \\ &\quad - \log \frac{b(z|y)}{\frac{1}{K}\sum_{k=1}^{K} b(z|y_k)} \end{aligned}$$

where (x, y) are sampled from the data distribution, $p(x, y)$, $e(z|x)$ is the learned *forward encoder* distribution, $b(z|y)$ is the learned variational *backward encoder* distribution, β is a Lagrange multiplier that controls how strongly compressed the learned representation Z is, with smaller β corresponding to less compression, and K is the number of examples in a mini-batch during training. We choose $e(z|x)$ and $b(z|y)$ to be parameterized von Mises-Fisher distributions as in [24].

C. Fleet Deployment

Once our policy achieves a satisfying performance in the *robot classrooms*, we deploy it in three real office buildings using a fleet of 23 robots. The robots use SLAM to localize themselves in the new building and navigate to a waste sorting station in their proximity. Once the robot is in front of the bin, we switch on our end-to-end RLS policy to perform the waste sorting task.

Since many of the waste sorting scenarios that we encounter in office buildings have not been seen previously and are likely to be seen only once due to the non-stationary nature of the task, and the overall throughput of data collection is limited (since people only deposit a limited amount of waste in the bins), we recreated the most commonly seen real-world scenarios in the *robot classrooms* to enable the robots to practice a realistic scenario for more trials. We periodically update these scenarios in the *robot classrooms* to reflect the most common waste sorting challenges encountered during the deployment. We use the data gathered during the deployment phase to train and improve all aspects of the RLS system: the classifier used for the object masks, RetinaGAN used for sim-to-real transfer as well as the PI-QT-Opt policy itself. To ensure that the most performant policy is deployed in office buildings at all times, we run the data flywheel described in Sec. V-C, which includes one week worth of data. While the bulk of our evaluations were conducted in the *robot classrooms* using scenarios recreated from real-world deployment (in order to allow systematic comparisons and repeatability), we discuss some performance measurements that we recorded during deployment in Section VI-E.

D. Neural network parameters and training

The critic network presented in Figure 3 has a total of 1.43M learnable parameters. The forward and backward encoder for

the predictive information loss each have 67k parameters. The architecture and parametrization largely follows [26]. In addition, the added LSTM is implemented by a ConvLSTM2D layer with 64 3×3 filters.

During training, we use Momentum optimizer with a learning rate of 0.0095 and momentum of 0.924; the batch size is 4096, training on a 4×4 slice of a TPUv3 pod (batch size 256 per chip). As described in Section V-C we chose a checkpoint from training by measuring OPC [12] on a held out dataset. To that end we train the model until OPC stabilizes, which typically happens well within the first 1 million training steps, and pick the checkpoint with the best OPC value (typically somewhere between 300k and 500k steps).

Overall, the training infrastructure for RLS consists of a large scale distributed system: 5000 simulator jobs continuously pick up new policy checkpoints and use it to gather new experience. 2000 log replay jobs read episodes collected on the real robots from disk. Both these data sources push episodes to a distributed set of 20 replay buffers. A separate set of jobs pulls from those buffers mixing real and simulated data at a ratio of 1:1, computes Q-value targets with the Bellman equation and trains new model checkpoints. In parallel, sets of simulators evaluate model performance in simulation, and another set of jobs calculates the OPC metric for each new checkpoint.

The stability of the data flow through this system is crucial for the performance of the resulting trained policy, but not trivial to achieve in a distributed system where every part can be randomly preempted due to resource constraints or machine failures. We thus spent considerable time fine-tuning it. We find that the most crucial part is to ensure that the training does not overfit to the experience in the replay buffers at any point in time, when data production slows down due to partial outages of the feeding jobs. The replay buffers are configured to hold a maximum of 2500 episodes, removing old episodes in FIFO manner when the buffers are full and additional data gets pushed. Sampling from the buffers occurs by drawing batches randomly across all samples currently in the buffer. To prevent overfitting we set a maximum re-use for each sample of 40 before we drop it from the buffer. Additionally we only allow sampling when the buffer contains at least 500 samples; otherwise the whole system waits until new data arrives.

E. Workspace safety constraints

Figure 11 shows a visualization of the workspace safety constraints for the robot's base and end-effector. Random samples around the robot's current pose are clipped to these constraints before being scored by the policy Q-value function in the CEM process.

F. Simulation details

1) RetinaGAN for visual domain transfer: As described in Section V-A, following previous work we employ RetinaGAN [10] to narrow the visual simulation-to-real domain gap, by applying an adapter GAN model to the images from our simulation. This approach greatly reduces the burden of creating high fidelity models and textures and the computational cost of complex rendering, allowing us to run our large scale simulations on machines without GPUs or other accelerators in the cloud. Figure 12 shows examples of images from our simulation adapted with RetinaGAN.

2) Deformable objects: For many of the objects we deal with, like cans and bottles, it is sufficient to model and simulate them as fully rigid, without introducing a large domain gap in the physical behavior. For others though, like for instance empty chips bags or other snack wrappers, we

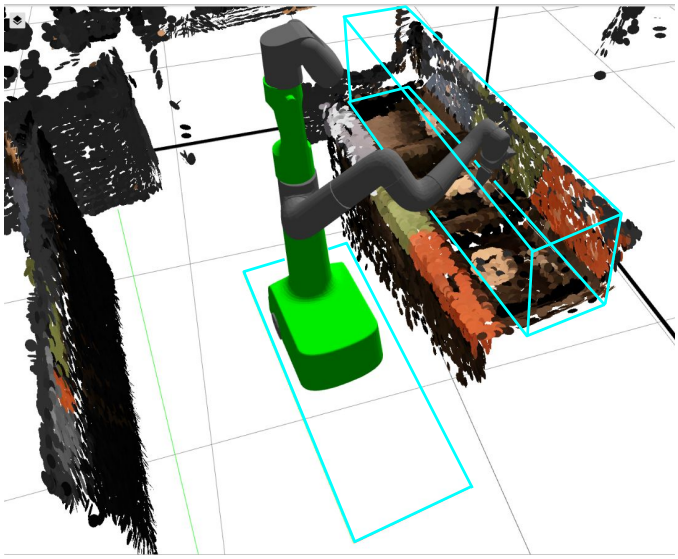

Fig. 11: A visualization of the robot and action-space boundaries. We span a box-constraint for the end-effector above the waste stations that allows the robot to explore sorting strategies while excluding non-productive poses. Similarly, for the base we define an area in front of the waste station that allows the robot to reach all parts of the waste station.

Fig. 12: Example 640x512 images generated using our simulator and adapted with RetinaGAN [10].

found it crucial to model and simulate them as deformable. If chips bags were modeled as rigid we would have to decide on a rigid shape. If we made them completely flat it would be near impossible to pick them up with our robot's gripper when lying flat in a tray, which is very unrealistic, since in reality it is enough to press down almost anywhere on the bag and pinch it to grasp. If we modeled them as crumpled up, it would be too easy to grasp them and the policy would again not learn the pressing down and pinching behavior required to get a good grip on a flattened out bag. Preliminary experiments without deformable objects showed that the policy, when facing a chips bag in simulation or in the real world would just immediately give up and terminate, "knowing" that it wouldn't be able to grasp them.

Figure 13 shows an animation of the robot interacting with deformable objects in our simulation.

G. Benchmark scenes taken from the "wild"

As described in the main paper, we create a set of scenarios representing actual situations our robots encountered at the deployment site (in the "wild") and use them for representative but repeatable experimentation in our *robot classrooms*. Fig. 4 shows images of the 9 in-distribution scenarios, Table 14 presents a description of the distribution of objects in the individual bins.

H. Object classes

We define a list of 12 classes of objects we encountered in the waste stations in the wild and used to define our benchmark scenes, along with a mapping of those object classes to waste target bin categories as follows:

- Recycle: "can", "bottle", "drink carton", "yogurt cup"
- Compost: "cup", "paper cup", "clear cup", "disposable bowl", "disposable plate", "paper container", "napkin"
- Landfill: "bag/wrapper", "face mask"

When continuing to run the policy at the deployment site, whenever we would see objects that don't belong to any of those categories, we would define new object classes for our annotation and metrics computation process. The list of those additional classes includes "non-disposable bowl", "non-disposable plate", "book/paper", "non-disposable drinkware", "food scraps", and "packaging". The frequency at which those objects appeared in the "wild" during the time of our experimentation did not warrant including them in our benchmark scenes.

I. Baseline performance of improved script

After bootstrapping our flywheel with the script (Alg. 1, line 1), we further invested engineering time to extend the initial script with various strategies specific to some objects that the initial version struggled with. For example, we had to modify the approach direction and gripper opening to grasp large lunch boxes more reliably. We, however, did not add data generated with the more specialized script to our flywheel, but instead let our policies continue to improve through trial-and-error. The goal of this experiment was to test the hypothesis

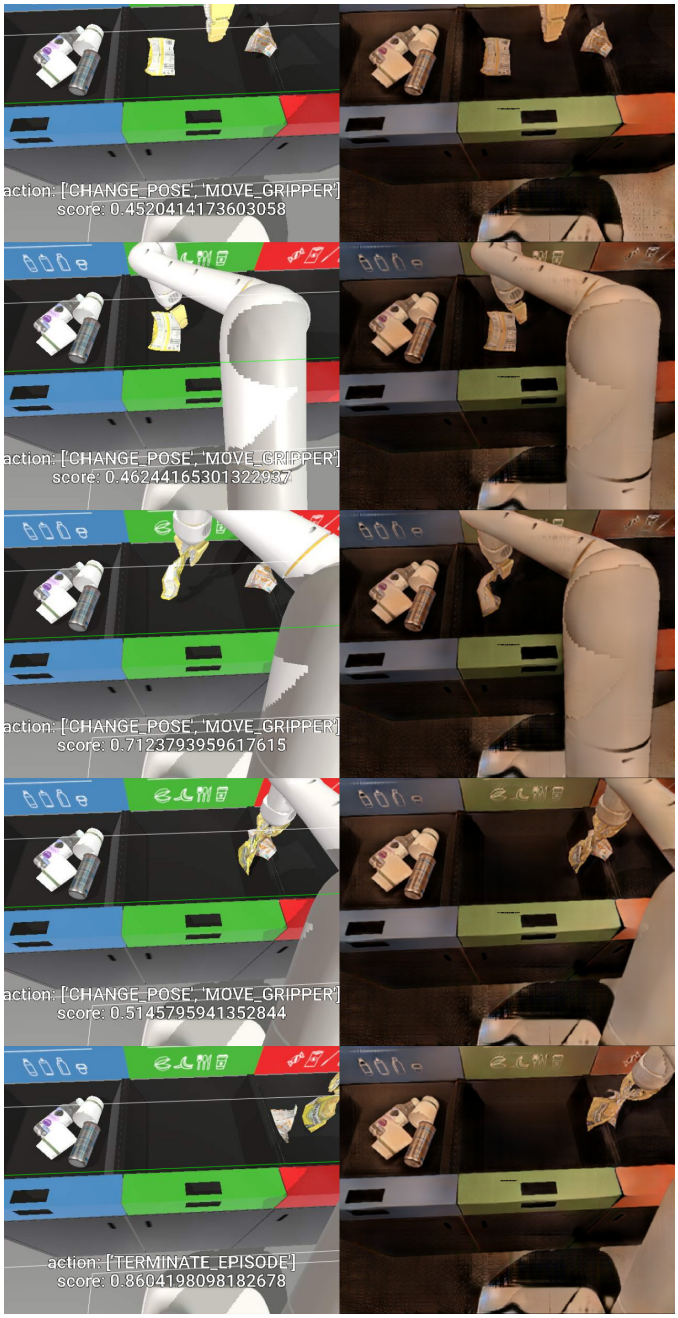

Fig. 13: Example of manipulating a deformable object (a chips bag) in simulation. Images on the left show the original rendering from our simulation, on the right are the corresponding adapted images using RetinaGAN [10]. The text on the left images shows the type of action chosen in the current step and the corresponding Q-value.

that performance gained through added engineering time can be met and surpassed with autonomous data-collection. We evaluate the extended script on the 9 waste scenarios we used for our evaluations in section VI. Overall, the improved script sorted 71% of the objects which is higher than our initial policy trained with only sim data. However, our learned policy further improved as we spun the flywheel without added engineering time and eventually reached a performance of

scene	recycle	compost	landfill
1	1 clear cup	1 cup, 2 disposable bowls, 2 napkins, 1 drink carton	
2	2 bottles, 6 cans	1 cup, 2 napkins	2 paper containers, 1 cup, 4 napkins
3	1 can, 1 bottle, 2 paper cup	2 bag/wrappers	
4	1 bottle, 1 clear cup, 1 paper cup	1 paper cup, 2 disposable bowls, 1 napkin, 2 bag/wrappers	1 bag/wrapper, 1 yoghurt, 1 disposable plate
5	1 bottle	1 paper cup, 1 bottle	2 clear cups, 1 disposable bowl
6	2 yogurt cups, 1 clear cup, 1 bag/wrapper, 2 napkins	2 clear cups, 4 paper cups, 4 napkins, 1 bag/wrapper	2 bottles, 2 napkins
7	1 bag/wrapper	2 paper cups, 1 clear cup, 1 paper bag	1 yogurt cup, 2 napkins
8	5 cans, 1 clear cup	1 disposable bowl, 3 paper cups, 1 clear cup, 4 napkins, 1 can, 1 bag/wrapper	1 paper container, 1 disposable bowl, 2 napkins
9	2 bottles, 3 cans, 2 bag/wrappers	1 paper container, 1 paper cup, 1 clear cup, 4 disposable bowls, 3 napkins, 1 yogurt cup	
held out 1	2 bottles, 8 cans	3 paper cups, 1 compostable bowl, 1 napkin	1 banana peel, 1 clear cup, 4 napkins, 2 containers
held out 2	1 bottle	1 bottle	2 clear cups, 1 compostable bowl, 1 packaging
held out 3	1 bag/wrapper	2 paper cups, 1 clear cup, 1 napkin, 1 face mask	2 napkins, 1 yogurt cup

Fig. 14: Description of the benchmark scenes used for evaluations, shown in Figure 4. A scene is defined by the initial quantity of objects of each class in each of the 3 waste bins. Misplaced objects are marked red.

84%. Although, it remains unclear how the script compares to a learned policy at the limit of increasingly more engineering time investment versus data collection, we observe that our data-driven approach outperforms a fair amount of engineering effort by a margin of 13 percentage points.