Bridge Data: Boosting Generalization of Robotic Skills with Cross-Domain Datasets

Frederik Ebert*[1], Yanlai Yang*[1] *, Karl Schmeckpeper[3], Bernadette Bucher[3], Georgios Georgakis[3], Kostas Daniilidis[3], Chelsea Finn[2], Sergey Levine[1] †

Abstract

Robot learning holds the promise of learning policies that generalize broadly. However, such generalization requires sufficiently diverse datasets of the task of interest, which can be prohibitively expensive to collect. In this paper, we ask: what would it take to enable practical data reuse in robotics for end-to-end skill learning? We hypothesize that the key is to use datasets with multiple tasks and multiple domains, such that a new user that wants to train their robot to perform a new task in a new domain can include this dataset in their training process and benefit from cross-task and cross-domain generalization. To evaluate this hypothesis, we collect a large multi-domain and multi-task dataset, with 7,200 demonstrations constituting 71 tasks across 10 environments, and empirically study how this data can improve the learning of new tasks in new environments. We find that jointly training with the proposed dataset and 50 demonstrations of a never-before-seen task in a new domain on average leads to a 2x improvement in success rate compared to using target domain data alone. We also find that data for only a few tasks in a new domain can bridge the domain gap and make it possible for a robot to perform a variety of prior tasks that were only seen in other domains.

1 Introduction

The prevailing paradigm of robot learning is to repeat data collection and policy training from scratch for every new task and environment. Learning policies in isolation not only increases the costs of data collection, but also limits the policy's scope of generalization.

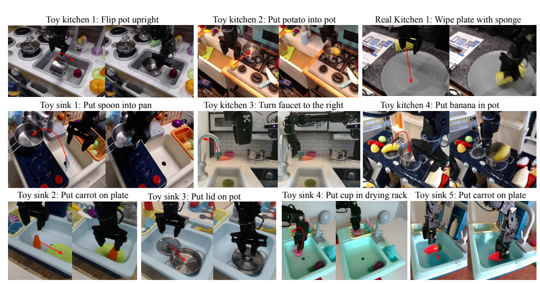

Figure 1: Illustration of our bridge dataset. The dataset includes demonstrations in 10 environments (4 toy kitchens and 5 toy sinks and 1 real kitchen), collected using a WidowX250 robot controlled via an Oculus Quest2 VR device, and consists of 7200 demonstrations. The red arrows indicate the desired movement of the target object.

In other fields, such as computer vision [14] and natural language processing (NLP) [3], utilizing large, diverse datasets has shown considerable success in enabling generalization to new problems or domains with a small amount of data (e.g., via pretraining and finetuning). However, in robotics, datasets are usually collected with a specific robotic platform and domain in mind, typically by the same researcher who intends to use that dataset. What would it take to make datasets reusable in robotics in the same way as large supervised datasets are reused (e.g., ImageNet [2])? Here we define a domain as the physical space a robot is situated in such as the type of (toy)kitchen, which implies not only different backgrounds but also different lighting conditions. The aim of our paper is to investigate the degree to which such a

*denotes equal contribution
†1 University of California Berkeley, 2 Stanford University, 3 University of Pennsylvania

NeurIPS 2021 Workshop on Robot Learning: Self-Supervised and Lifelong Learning, Virtual, Virtual

multi-task and multi-domain dataset, which we refer to as a *bridge* dataset, can enable a new robot in a new domain (which was not seen in the bridge data) to more effectively generalize when learning a new task (which was also not seen in the bridge data), as well as to transfer tasks from the bridge data to the target domain. We also propose a new dataset that enables this goal in the context of kitchen-themed tasks with a low-cost robotic arm and is intended to be reused by other researchers.

No existing dataset covers *both* multiple tasks *and* multiple domains in a way that is suitable to study our central hypothesis: can prior data be used to improve the generalization of *new* tasks in *new* domains? We will call this the *bridge data hypothesis*. We believe this is a critical requirement for effective data reuse in robotics, where different labs and researchers can all bootstrap from the same shared datasets. We present our new dataset, and then use it to evaluate the bridge data hypothesis that is stated above, on three types of transfer scenarios that are outlined in more detail in section 3.

The main contributions of our work consist of an empirical evaluation of the bridge data hypothesis and a practical example of a bridge dataset with 7,200 demonstrations for 71 tasks in 10 environments, which we have released publicly on the project website.[3] Our results suggest that accumulating and reusing diverse multi-task and multi-domain datasets, at least when all data is collected with the same type of robot, may make it possible for researchers to endow robots with generalizable skills using only a modest amount of in-domain data for their desired task.

Dataset	# Tasks	# Trajec.	# Domains
DAML [25]	3	2.9k	1
MIME [22]	22	8.2k	1
RoboNet [1]	N/A	162k	7
RoboTurk [19, 18]	3	2.1k	1
Vis. Imit. Made [24]	2	2k	50
Ours	**71**	7.2k	10

Figure 2: Comparison of our dataset and prior works. Our dataset has by far the most tasks, and is the only dataset with more than 2 tasks that has many domains. This is critical for evaluating the bridge data hypothesis.

section 10 will explain our low cost data collection system in more detail.

2 Related Work

While most prior work on deep visuomotor learning trains a single task in a single domain [8, 4, 11, 25, 17, 21, 9, 23, 27], our goal is not to develop better learning methods, but rather to illustrate how generic multi-domain, multi-task datasets can be used with existing algorithms to boost the generalization of *new* tasks in *new* domains. Prior work on multi-task reinforcement learning [13] has shown that data from other tasks can boost generalization of new tasks, however this study is carried out in a *single* domain.

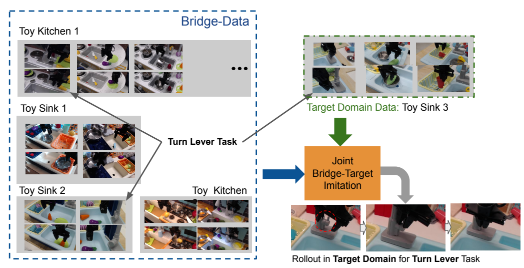

Figure 3: **Scenario (1): transfer with matching behaviors**. In this setting, bridge data is used to improve the performance and generalization of tasks in the target domain for which the user has collected some amount of data. These tasks must also be present in the bridge data. In this example, the user demonstrates the "turn lever," "squash into pot," and "flip cup" tasks in the target domain, and these tasks are also present in several domains in the bridge data. After including the bridge data in training, the performance and generalization of these tasks in the target is significantly higher.

Existing robot learning datasets do not exhibit the right properties for boosting the generalization of new tasks in *new* domains or zero-shot transferring skills from the prior dataset to a target domain. We provide an overview of the most related datasets in Figure 2. Most existing robot datasets, such as MIME [22], DAML [25], RoboTurk [19, 18], and many others [20, 6, 16, 12, 5, 26, 13] only feature a single domain, making them difficult to use for boosting the generalization in *other* domains. Merging multiple existing datasets into one multi-domain dataset is difficult due to inconsistencies in data collection protocols, time discretization, robot morphologies, and sensors.

3 Boosting Generalization via Bridge Datasets

We study how a bridge dataset can be used to boost three types of generalization, though other modes may also be feasible:

[3] https://sites.google.com/view/bridgedata

(1) Transfer with matching behaviors, where the user collects some small amount of data in their target domain for tasks that are also present in the bridge data (e.g., around 50 demos per task), and uses the bridge data to boost the performance and generalization of these tasks. We illustrate this scenario in Figure 3. This scenario is the most conventional, and resembles domain adaptation in computer vision, but it is also the most limiting, since it requires the user's desired tasks to be present in the bridge data.

(2) Zero-shot transfer with target support, where the user utilizes data from a few tasks in their target domain to "import" other tasks that are present in the bridge data *without* additionally collecting new demonstrations for them in the target domain. For example, the bridge data contains the tasks of putting a sweet potato into a pot or a pan, the user provides data in their domain for putting brushes in pans, and the robot is then able to *both* put brushes as well as put sweet potatoes in pans. We illustrate this scenario in Figure 4. This scenario increases the repertoires of skills that are available in the user's target environment, simply by including the bridge data, thus eliminating the need to recollect data for every task in every target environment.

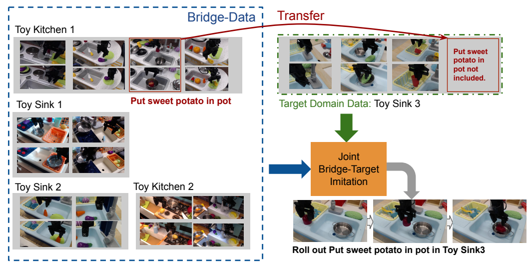

Figure 4: **Scenario (2): zero-shot transfer with target support**. In this setting, the goal is to "import" a task from the bridge data that was *not* seen in the target domain. The user provides a few tasks in the target domain that are used to connect to the bridge data, and then asks the robot to perform a task that they did not provide, but which was seen in the bridge data. In this case, the "put sweet potato in pot" task is present in the toy kitchen 1 domain in the bridge data, but is *not* demonstrated by the user in the target domain. After training with user-provided data for other tasks, the robot is able to perform "put sweet potato in pot" in the target domain.

(3) Boosting generalization of new tasks, where the user provides a small amount of data (50 demonstrations in practice) for a new task that is not present in the bridge data, and then utilizes the bridge data to boost generalization and performance of this task. This scenario, illustrated in Figure 5, most directly reflects our primary goals, since it uses the bridge data without requiring *either* the domains or tasks to match, leveraging the diversity of the data and structural similarity to boost performance and generalization of entirely new tasks.

4 Using Bridge Data in Imitation Learning

As a proof-of-concept to illustrate the utility of bridge datasets for boosting generalization in robot learning, we will present experimental results for an imitation-based approach (please find details in the appendix) that utilizes this data, although the data could also be used with a variety of other robotic learning algorithms such as offline RL and model-based planning. While a variety of transfer learning methods have been proposed in the literature for combining datasets from distinct domains, we found that a simple joint training approach is effective for deriving considerable benefit from bridge data. For each of the scenarios outlined in Section 3, we take the user-provided demonstrations in the target domain and combine them with the entire bridge dataset for training. Training details are provided in section 8 and an explanation of the policy architecture is given in section 7 in the appendix.

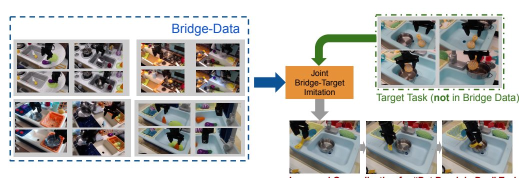

Figure 5: **Scenario (3): boosting generalization of new tasks**. The user provides some data for a *new* task that was not seen in the bridge data, and the bridge data is included in training to boost performance and generalization for this new task.

5 Experimental Results

Our experimental evaluation aims to study how well bridge data can facilitate generalization in scenarios **(1)**, **(2)**, and **(3)**, as outlined in Section 3. We evaluate generalization on a set of new target domains with limited target domain data for each of the generalization scenarios, and compare the performance of learned policies with and without bridge data. Videos of the experiments are included in the supplementary materials and on the project webpage. Please see the appendix for a description of our quantitative evaluation metrics.

Scenario (1): transfer with matching behaviors. Figure 6 (left) shows results for the transfer learning with matching behaviors scenario, where the user provides some data for a set of tasks in the target domain (which are also present in the bridge data), and we evaluate whether including bridge data during training improves performance and generalization. For comparison, we include the performance of the policy when trained *only* on the target domain data, without bridge data (Target Domain Only), a baseline that uses only the bridge data without any target domain data (Direct Transfer), as well as baseline that trains a single-task policy on data in the target domain only (Single Task). As can be seen in the results, jointly training with the bridge data leads to significant gains in performance (66% success averaged over tasks) compared to the direct transfer (14% success), target domain only (28% success) and the single task (18% success) baseline.

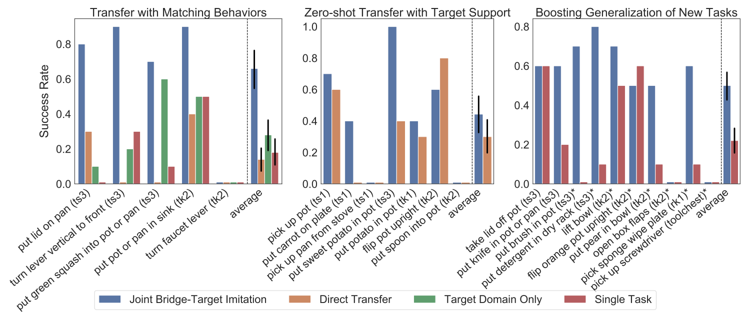

Figure 6: Comparisons of joint training with bridge data (blue) and other approaches for each type of scenario. The black vertical lines on the average success rate bar denote the standard error of the mean across different tasks for that scenario. **Left:** Performing joint training on bridge and target data leads to improved performance, here the task is included both in the bridge and target dataset. **Middle:** Using target domain data from other tasks helps transferring tasks from the bridge dataset to the target domain. **Right:** Joint training with the bridge data and a target task that is *not* contained in the bridge dataset enables significant generalization improvement compared to only training on the target task alone. Tasks with an asterisk (*) uses objects that are not part of the bridge dataset.

Scenario (2): zero-shot transfer with target support. We provide a qualitative example for this scenario in Figure 7 middle (in the appendix), which shows an experiment where we transfer the "put carrot on plate" task into the Toy Sink 1 target domain using the bridge data and target domain data consisting of 10 *other* tasks. The results, shown in Figure 6 (middle), indicate that the jointly trained policy which obtains 44% success averaged over tasks indeed attains a very significant increase in performance over direct transfer (30% success), suggesting that the zero-shot transfer with target support scenario offers a viable way for users to "import" tasks from the bridge dataset into their domain.

Scenario (3): boosting generalization of new tasks. We collected data for 10 different unique tasks in 4 different environments and excluded them from the bridge data to simulate a user collecting their own unique task in their new target environment. Figure 7 right (in the appendix) illustrates one of these scenarios, where we collected 50 demonstrations for the "wipe place with sponge" task in the the real kitchen 1 target domain. *Neither* data from the target domain nor this task or this object are present in the bridge data. After jointly training with both bridge and target data we obtain a significant generalization boost when running the policy in the target domain, compared to a policy trained on only the single-task target domain data. The results are presented in Figure 6 (right), and show that training jointly with the bridge data leads to significant improvement on 6 out of 10 tasks across three evaluation environments, leading to 50% success averaged over tasks, whereas single task policies attain around 22% success – a 2× improvement in overall performance.

6 Conclusion

We show how a large, diverse bridge dataset can be leveraged to improve generalization in robotic learning. Our experiments demonstrate that including bridge data when training skills in a new domain can improve performance across a range of scenarios, both for tasks that are present in the bridge data and, perhaps surprisingly, entirely new tasks. This means that bridge data may provide a generic tool to improve generalization in a user's target domain. In addition, we showed that bridge data can also function as a tool to *import* tasks from the prior dataset to a target domain, thus increasing the repertoires of skills a user has at their disposal in a particular target domain. This suggests that a large, shared bridge dataset, like the one we have released, could be used by different robotics researchers to boost the generalization capabilities and the number of available skills of their imitation-trained policies.

References

[1] 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.

[2] 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*, 2009.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] Yan Duan, Marcin Andrychowicz, Bradly C Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. *arXiv preprint arXiv:1703.07326*, 2017.

[5] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

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

[7] Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep spatial autoencoders for visuomotor learning. In *International Conference on Robotics and Automation (ICRA)*, 2016.

[8] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *Conference on Robot Learning*, pages 357–368. PMLR, 2017.

[9] Ali Ghadirzadeh, Xi Chen, Wenjie Yin, Zhengrong Yi, Marten Bjorkman, and Danica Kragic. Human-centered collaborative robots with deep reinforcement learning. *IEEE Robotics and Automation Letters*, 2020.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, 2016.

[11] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *arXiv preprint arXiv:1606.03476*, 2016.

[12] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pages 651–673. PMLR, 2018.

[13] 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 preprint arXiv:2104.08212*, 2021.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[15] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[16] 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.

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

[18] Ajay Mandlekar, Jonathan Booher, Max Spero, Albert Tung, Anchit Gupta, Yuke Zhu, Animesh Garg, Silvio Savarese, and Li Fei-Fei. Scaling robot supervision to hundreds of hours with roboturk: Robotic manipulation dataset through human reasoning and dexterity. *arXiv:1911.04052*, 2019.

[19] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, Silvio Savarese, and Li Fei-Fei. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, 2018.

[20] 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 (ICRA)*. IEEE, 2016.

[21] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1134–1141. IEEE, 2018.

[22] 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.

[23] Stephen Tian, Suraj Nair, Frederik Ebert, Sudeep Dasari, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. Model-based visual planning with self-supervised functional distances. *arXiv preprint arXiv:2012.15373*, 2020.

[24] Sarah Young, Dhiraj Gandhi, Shubham Tulsiani, Abhinav Gupta, Pieter Abbeel, and Lerrel Pinto. Visual imitation made easy. *arXiv e-prints*, pages arXiv–2008, 2020.

[25] Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *arXiv preprint arXiv:1802.01557*, 2018.

[26] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36(4):1307–1319, 2020.

[27] 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.

Figure 7: Examples of successful trajectories performed by the policy jointly trained with prior data and target domain data. Left: put pot in sink (scenario 1); middle: put carrot on plate (scenario 2); Right: wipe plate with sponge (scenario 3).

7 Policy architecture.

We use task-conditioned behavioral cloning (BC) with an additional task-id input to the policy, which is used to distinguish tasks during training and testing. In some cases, a task cannot be uniquely determined by only observing the input image, and a one-hot vector representing the task will solve this issue. The images are first fed into a 34-layer ResNet [10] and the resulting feature maps are passed through a spatial softmax [7, 15], which extracts a set of spatial positions of the relevant features. The spatial features are then concatenated with the one-hot task-id vector, and are fed into 3 layers of fully-connected networks by which the final action prediction is produced. During training, for a batch of training data containing tuples of task ids, images, and ground-truth actions, the network is trained by minimizing the standard ℓ_2-error between the ground-truth actions and the predicted actions given by the policy provided the task id and the image observation as the input.

8 Training details.

Since the amount of target domain data is usually significantly less than the amount of bridge data, we rebalance the two datasets during training. In the matching behaviors and zero-shot transfer with target support scenarios, the ratio between the number of trajectories in the bridge and target data is roughly 10:1, and we rebalance the data such that 70% of the dataset is bridge data and 30% is target domain data. In the “boosting generalization of new tasks" scenario the imbalance is more severe, roughly 60:1, and so we rebalance such that 90% of the dataset is bridge data and 10% is target domain data. Lower rebalancing ratios of bridge data and target domain data tend to produce overfitting when the amount of target domain data is as low as 50 demonstrations.

Task	Tar. Env	Joint Train	Single Task	Potential reason for no gain with bridge data
turn faucet lever (1)	tk2	0%	0%	The faucet in tk2 has very different appearance from the other faucets
Pickup pan from stove (2)	ts1	0%	N/A	Not enough target domain data and prior data for this task
Put spoon into pot (2)	tk2	0%	N/A	Not enough target domain data and prior data for this task
flip orange pot upright (3)	tk2	50%	60%	There is no orange pot in prior dataset, only metal pots in prior dataset
open box flaps (3)	tk2	10%	10%	Boxes and pushing motions do not occur in prior data
take lid off pot (3)	ts3	60%	60%	Only 100 demos involving lids in prior dataset.
pick up screw driver (3)	toolchest	0%	0%	The toolchest and screwdrivers are visually very different from prior data
put pot or pan in sink (1)	tk2	90%	50%	
put carrot on plate (2)	ts1	40%	N/A	
Wipe plate w/ sponge (3)	k1	70%	N/A	
put pear in bowl (3)	tk2	50%	10%	
put brush in pot (3)	ts3	90%	0%	
put detergent dry rack (3)	ts3	80%	10%	
lift bowl (3)	tk2	70%	50%	

Figure 8: Comparison of scenarios where usage of the bridge data helps performance and where it does not. Scenarios where usage of bridge data does not help are marked in red font. The type of transfer setting is denoted by the number in brackets after the task description.

9 Quantitative metrics.

All quantitative evaluations use 10 trials per task,varying object positions and distractors on every trial and varying the position of the robot relative to the environment every 5 trials. This ensures that all test configurations are unique and different from any condition seen in training, providing a measurement of generalization performance for the policy. When the experiments in toy kitchen 1-3 and toy sink 1-3 were conducted, the bridge dataset only comprised 4700 trajectories. Other experiments use the full dataset with 7200 trajectories total.

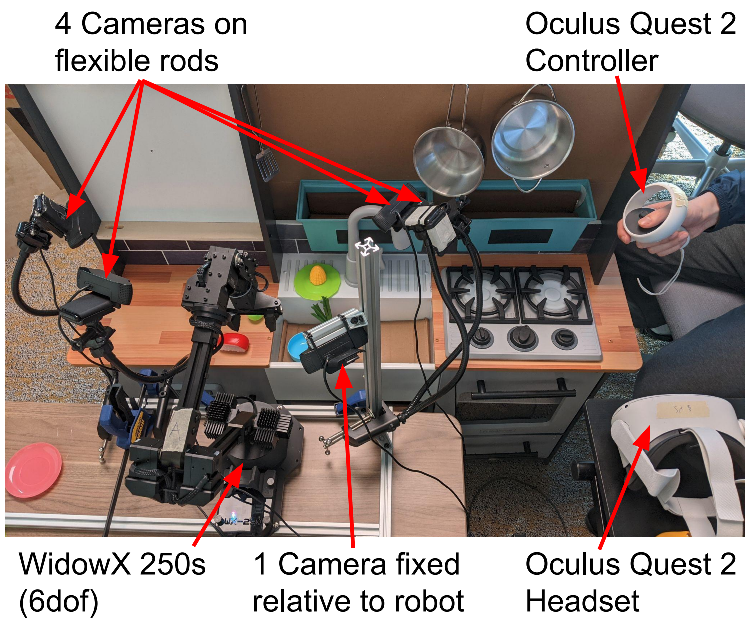

Figure 9: Demonstration data collection setup using VR Headset. The scene is captured by 5 cameras simultaneously. While one of the cameras is fixed, the others are mounted on flexible rods.

10 Robotic system overview

Since our dataset is likely the most useful for users with the same or similar type of robot, we chose to use a low-cost and widely available robot, a 6-dof WidowX250s (US$2900), which many other users of our dataset are likely to be able to obtain. The total cost of the setup is less than US$3600 (excluding the computer). To collect demonstrations, we use an Oculus Quest headset, where we put the headset on a table as illustrated in Figure 9 next to the robot and track the user's handset while applying the user's motions to the robot end-effector via inverse kinematics. We capture images from 3 to 5 cameras concurrently, using standard webcams as well as Intel RealSense depth cameras.

11 Task List

task number	entity	domain	task	number of demos
1	berkeley	realkitchen1_counter	put_spoon_on_plate	16
2	berkeley	realkitchen1_counter	pick_up_sponge_and_wipe_plate	50
3	berkeley	toysink2_bww	flip_pot_upright_which_is_in_sink	50
4	berkeley	toysink2_bww	put_cup_from_counter_or_drying_rack_into_sink	50
5	berkeley	toysink2_bww	put_carrot_on_plate	54
6	berkeley	toysink2_bww	put_knife_on_cutting_board	50
7	berkeley	toysink2_bww	turn_lever_vertical_to-front	50
8	berkeley	toysink2_bww	put_spoon_in_pot	50
9	berkeley	toysink2_bww	put_eggplant_into_pot_or_pan	50
10	berkeley	tool_chest	pick_up_violet_Allen_key	50
11	berkeley	tool_chest	pick_up_blue_pen_and_put_into_drawer	50
12	berkeley	tool_chest	pick_up_closest_rainbow_Allen_key_set	5
13	berkeley	tool_chest	pick_up_scissors_and_put_into_drawer	50
14	berkeley	tool_chest	pick_up_bit_holder	50
15	berkeley	tool_chest	pick_up_red_srewdriver	50
16	berkeley	tool_chest	pick_up_glue_and_put_into_drawer	35
17	berkeley	tool_chest	pick_up_box_cutter_and_put_into_drawer	50
18	berkeley	toykitchen4	put_detergent_in_sink	50
19	berkeley	toykitchen4	put_carrot_in_bowl	33
20	berkeley	toykitchen4	put_lid_on_pot_or_pan	50
21	berkeley	toykitchen4	put_pear_on_plate	50
22	berkeley	toykitchen4	put_banana_in_pot_or_pan	50
23	berkeley	toykitchen4	put_sushi_in_pot_or_pan	50
24	berkeley	toysink1_room8052	put_spoon_into_pan	50
25	berkeley	toysink1_room8052	flip_pot_upright_which_is_in_sink	50
26	berkeley	toysink1_room8052	put_pan_from_drying_rack_into_sink	50
27	berkeley	toysink1_room8052	put_eggplant_into_pan	51
28	berkeley	toysink1_room8052	put_pan_on_stove_from_sink	50
29	berkeley	toysink1_room8052	put_pan_from_sink_into_drying_rack	50
30	berkeley	toysink1_room8052	put_pan_from_stove_to_sink	50
31	berkeley	toysink3_bww	put_knife_in_pot_or_pan	50
32	berkeley	toysink3_bww	put_cup_into_pot_or_pan	50
33	berkeley	toysink3_bww	put_lid_on_pot_or_pan	50
34	berkeley	toysink3_bww	put_green_squash_into_pot_or_pan	50
35	berkeley	toysink3_bww	turn_lever_vertical_to_front	50
36	berkeley	toysink3_bww	put_cup_from_anywhere_into_sink	50
37	berkeley	toysink3_bww	flip_cup_upright	50
38	berkeley	toysink3_bww	put_brush_into_pot_or_pan	50
39	berkeley	toysink3_bww	put_pot_or_pan_from_sink_into_drying_rack	50
40	berkeley	toysink3_bww	put_detergent_from_sink_into_drying_rack	50
41	berkeley	toysink3_bww	take_lid_off_pot_or_pan	50
42	berkeley	toykitchen2_room8052	put_corn_on_plate	50
43	berkeley	toykitchen2_room8052	put_sweet_potato_in_pot	50
44	berkeley	toykitchen2_room8052	put_lemon_on_plate	50
45	berkeley	toykitchen2_room8052	flip_orange_pot_upright_in_sink	50
46	berkeley	toykitchen2_room8052	flip_salt_upright	6
47	berkeley	toykitchen2_room8052	put_knife_on_cutting_board	15
48	berkeley	toykitchen2_room8052	put_sushi_on_plate	50
49	berkeley	toykitchen2_room8052	put_pot_or_pan_on_stove	49
50	berkeley	toykitchen2_room8052	turn_lever_vertical_to_front	50
51	berkeley	toykitchen2_room8052	put_potato_in_pot_or_pan	50
52	berkeley	toykitchen2_room8052	put_spatula_in_pan	50
53	berkeley	toykitchen2_room8052	put_strawberry_in_pot	50
54	berkeley	toykitchen2_room8052	put_carrot_in_pot_or_pan	50
55	berkeley	toykitchen2_room8052	put_pear_in_bowl	50
56	berkeley	toykitchen2_room8052	put_potato_on_plate	99
57	berkeley	toykitchen2_room8052	put_can_in_pot	50
58	berkeley	toykitchen2_room8052	lift_bowl	50
59	berkeley	toykitchen2_room8052	put_pot_or_pan_in_sink	51
60	berkeley	toykitchen2	open_small4fbox_flaps	53
61	berkeley	toykitchen2	open_brown1fbox_flap	50
62	berkeley	toykitchen2	close_small4fbox_flaps	25

63	berkeley	toykitchen2	open_white1fbox_flap	50
64	berkeley	toykitchen2	close_white1fbox_flap	50
65	berkeley	toykitchen2	close_brown1fbox_flap	51
66	berkeley	toykitchen1	lever_vertical_to_front	169
67	berkeley	toykitchen1	put_corn_into_bowl	50
68	berkeley	toykitchen1	put_sweet_potato_in_pan_which_is_on_stove_distractors	26
69	berkeley	toykitchen1	put_sweet_potato_in_pan_which_is_on_stove	25
70	berkeley	toykitchen1	put_detergent_in_sink	50
71	berkeley	toykitchen1	pick_up_pot_from_sink_distractors	100
72	berkeley	toykitchen1	put_pepper_in_pan	50
73	berkeley	toykitchen1	put_broccoli_in_pot_cardboardfence	130
74	berkeley	toykitchen1	put_eggplant_on_plate	50
75	berkeley	toykitchen1	take_sushi_out_of_pan	51
76	berkeley	toykitchen1	put_pepper_in_pot_or_pan	100
77	berkeley	toykitchen1	take_carrot_off_plate	50
78	berkeley	toykitchen1	take_broccoli_out_of_pan	50
79	berkeley	toykitchen1	put_red_bottle_in_sink	50
80	berkeley	toykitchen1	open_large4fbox_flaps	51
81	berkeley	toykitchen1	turn_faucet_front_to_left	117
82	berkeley	toykitchen1	put_lid_on_pot_or_pan	100
83	berkeley	toykitchen1	pick_up_pan_from_stove_distractors	80
84	berkeley	toykitchen1	put_carrot_on_plate	100
85	berkeley	toykitchen1	put_corn_in_pan_which-is_on_stove_distractors	26
86	berkeley	toykitchen1	put_pan_in_sink	51
87	berkeley	toykitchen1	close_small4fbox_flaps	27
88	berkeley	toykitchen1	put_big_spoon_from_basket_to_tray	54
89	berkeley	toykitchen1	take_can_out_of_pan	2
90	berkeley	toykitchen1	put_knife_on_cutting_board	50
91	berkeley	toykitchen1	put_carrot_on_cutting_board	45
92	berkeley	toykitchen1	put_corn_in_pot_which_is_in_sink_distractors	100
93	berkeley	toykitchen1	close_large4fbox_flaps	51
94	berkeley	toykitchen1	put_small_spoon_from_basket_to_tray	54
95	berkeley	toykitchen1	put_sushi_on_plate	50
96	berkeley	toykitchen1	put_broccoli_in_bowl	25
97	berkeley	toykitchen1	put_sweet_potato_in_pot_which_is_in_sink_distractors	100
98	berkeley	toykitchen1	twist_knob_start_vertical_clockwise90	9
99	berkeley	toykitchen1	put_eggplant_in_pot_or_pan	100
100	berkeley	toykitchen1	turn_lever_vertical_to_front_distractors	330
101	berkeley	toykitchen1	put_pot_on_stove_which_is_near_stove_distractors	102
102	berkeley	toykitchen1	put_pot_in_sink	50
103	berkeley	toykitchen1	put_lid_on_stove	51
104	berkeley	toykitchen1	pick_up_bowl_and_put_in_small4fbox	50
105	berkeley	toykitchen1	put_green_squash_in_pot_or_pan	46
106	berkeley	toykitchen1	put_broccoli_in_pot_or_pan	75
107	berkeley	toykitchen1	put_banana_on_plate	50
108	berkeley	toykitchen1	put_pear_in_bowl	50
109	berkeley	toykitchen1	flip_pot_upright_in_sink_distractors	306
110	berkeley	toykitchen1	put_corn_in_pan_which_is_on_stove_distractors	131
111	berkeley	toykitchen1	take_lid_off_pot_or_pan	50
112	berkeley	toykitchen1	put_fork_from_basket_to_tray	58
113	berkeley	realkitchen1_dishwasher	pick_up_any_cup	50
114	berkeley	realkitchen1_dishwasher	pick_up_glass_cup	61
115	berkeley	realkitchen1_dishwasher	pick_up_green_mug	49
116	upenn	toykitchen3	turn_faucet_right_55	55
117	upenn	toykitchen3	pick_up_pot_50	50
118	upenn	toykitchen3	turn_faucet_left_56	56
119	upenn	toysink5	turn_lever_vertical_to_front	106
120	upenn	toysink5	put_cup_from_anywhere_into_sink	104
121	upenn	toysink5	flip_cup_upright	111
122	upenn	toysink4	put_cup_from_sink_to_drying_rack	25
123	upenn	toysink4	move_faucet_front_to_left	9
124	upenn	toysink4	put_cup_from_drying_rack_to_sink	16
125	upenn	toysink4	put_eggplant_on_plate	25
126	upenn	toysink4	put_carrot_on_plate	25
127	upenn	toysink4	put_cup_from_counter_to_sink	101
128	upenn	toysink4	turn_faucet_front_to_right	25
total demos	7453			